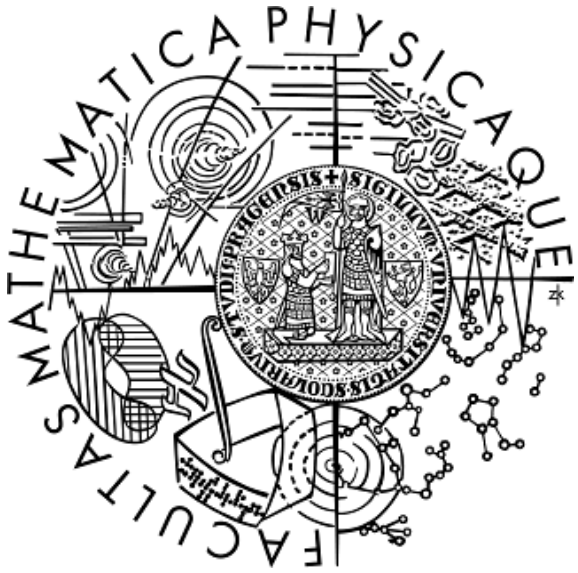
Univerzita Karlova v Praze
Matematicko – fyzikální fakulta

BAKALÁRSKA PRÁCA



Peter Lapin

Minority Report

Kabinet software a výuky informatiky

Vedúci Bakalárskej práce: RNDr. Josef Pelikán

Študijný program: Informatika, obecná informatika

2008

Chcel by som sa poďakovať v prvom rade svojim rodičom, ktorí ma pri mojom štúdiu podporovali a vedúcemu tejto práce, pánovi RNDr. Josefovi Pelikánovi za odbornú pomoc a čas, ktorý mi venoval.

Prehlasujem, že som svoju bakalársku prácu napísal samostatne a výhradne s použitím citovaných prameňov. Súhlasím s požičiavaním práce a jej zverejňovaním.

V Prahe dňa 18.7.2008 Peter Lapin

Obsah

Názov práce: Minority Report
Autor: Peter Lapin
Katedra: Kabinet software a výuky informatiky
Vedúci bakalárskej práce: RNDr. Josef Pelikán
e-mail: Josef.Pelikan@mff.cuni.cz

Abstrakt: V predloženej práci študujeme možnosti návrhu aplikácie na ovládanie kurzoru myši pomocou lacnej webkamery a špeciálne upravenej rukavice s pripevnenými LED diódami. Diódy sú umiestnené na palci, ukazováku a prostredníku. Ich farby sú po rade červená, modrá a zelená. Pojednávame o algoritmoch použitých na rozpoznanie diód v zachytenej video sekvencií obrazu. Ďalej o návrhu a rozopoznaniu gest rukou, tak aby bolo možné úplne zastúpiť funkcionalitu myši. A napokon rozoberáme algoritmy použité pre ovládanie kurzoru myši. V rámci tejto práce bola vyvinutá aplikácia, ktorá je po spustení, schopná úplne nahradiť myš.

Kľúčové slová: rozpoznávanie, gestá rukou, pohyb myši.

Title: Minority Report
Author: Peter Lapin
Department: Department of software and computer science education
Supervisor: RNDr. Josef Pelikán
Supervisor's e-mail address: Josef.Pelikan@mff.cuni.cz

Abstract: In this thesis we study the possibilities of designing an application to control the mouse with a cheap webcam and a special glove with 3 light emitting diodes (LEDs). The diodes are placed on the thumb, the index finger and the midddle finger. We discuss the algorithms, used for diode recognition in a video stream. Then we propose hand gestures, so the glove and a webcam can fully substitute the mouse functionality. At last we talk about algorithms for moving the mouse cursor. As result of our work a software was developed, which is able to fully substitute the mouse.

Keywords: recognition, hand gestures, mouse movement

Úvod

Inšpiráciou pre túto prácu bol film „Minority Report“ [1] v ktorom hlavný hrdina ovláda superpočítač budúcnosti za pomoci dvoch rukavíc s LED diódami na palcoch ukzovákoch a prostredníkoch. Dnešné počítače, ako každý vie, sa ovládajú za pomoci klávesnice a myši. Cieľom tejto práce bolo nahradiť myš, a jej kurzorom pohybovať bez toho, aby sme ju museli držať v ruke. Bola teda vytvorená rukavica s troma farebnými diódami umiestnenými na palci, ukazováku a prostredníku. Po nasadení na ruku je snímaná lacnou webkamerou a za pomoci dohodnutých gest rukou sa emuluje pohyb kurzora myši, klikanie tlačidiel, a skrolovanie. Diódy sú rôznofarebné. Pre jednoduchšiu identifikáciu boli vybrané farby červená (palec) modrá (ukazovák) a zelená (prostredník).

Úloha bola rozdelená na menšie podúlohy tak, aby sa každá z nich dala riešiť samostatne a nezávisle na druhej. Z technického hľadiska bolo potrebné najprv zachytiť obraz z kamery a dostať do počítača vo vhodnej reprezentácií. Nielen k tomuto, ale aj k ďalším účelom nám poslúžila knižnica OpenCV [2]. Nakoniec bolo potrebné, pre už vyriešenú úlohu, navrhnúť jednoduché užívateľské prostredie. O riešení týchto dvoch technických problémov sa môže čitateľ dozvedieť v priloženej programátorskej dokumentácií. Z logického hľadiska sa úloha rozdelila na jednotlivé celky nasledovným spôsobom. Prvým je identifikácia farebných diód na jednotlivých obrázkoch z video sekvencie. Druhý problém je navrhnutie „šikovných“ gest rukou, na ktorý nadväzuje problém ich rozpoznávania. Štvrtým problémom je vykonávanie akcií a generovanie udalostí myši. Vďaka takémuto rozdeleniu sme mohli testovať jednotlivé podúlohy v rôznych kombináciách zapojení za sebou a tým sa vývoj celej aplikácie urýchlil. Pre testovanie jednotlivých algoritmov bol vytvorený „framework“, v ktorom sa dali za behu aplikácie jednotlivé algoritmy prepínať. Mohli sme teda hneď vizuálne kontrolovať výsledky výskumu. V nasledujúcich kapitolách sa budeme jednotlivými riešeniami problémov zaoberať. Povieme si o dobrých i zlých nápadoch na ich riešenie.

1Úvod do počítačového videnia

Predstava vytvorenia inteligentného stroja fascinuje ľudí už dlhé roky. Ako prvý, ktorý sa pokúsil o akúsi klasifikáciu strojov bol Turing v roku 1950. Od tej doby je predstava skúmaná pracovníkmi v oblasti umelej inteligencie, a počítačového videnia. Ich snahou je „obdariť" počítače schopnosťou spracovávania informácií a ich porovnávaniu tak, ako to dokážu biologické organizmy.

Počítačové videnie sa zaoberá explicitnou konštrukciou zmysluplného popisu fyzických objektov na obrázku. [3] Je to ešte relatívne nový odbor, ale zato rýchlo sa rozvíjajúci. Výsledky výskumu sa využívajú v rôznych odvetviach. Napríklad priemysle, zdravotníctve atd.

V nasledujúcich podkapitolách si preberieme nutné pojmy, na ktoré sa budeme v priebehu práce odkazovať.

1.1Základné pojmy

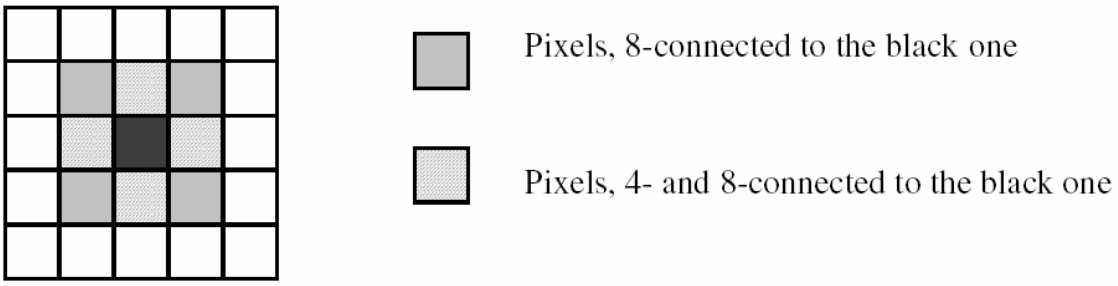
Obrázok v odtieňoch šedej[1] **–** obrázok, ktorého každý pixel obsahuje jednu hodnotu, ktorá je informáciou o intenzite pixelu [4]

Binárny obrázok – Skladá sa len z 0-pixelov a 1-pixelov, pričom prvé majú hodnotu 0 a druhé hodnotu 1. [5]

0-(1-) komponenta – Množina pospájaných 0- alebo 1-pixelov na obrázku. [5]

4-spojitosť – Vlastnosť dvoch pixelov so súradnicami (x, y) a $(\bar{x}, \bar{y})$. Dva pixely sú 4-spojité vtedy a len vtedy, ak platí $|x - \bar{x}| + |y - \bar{y}| = 1$. [5]

8-spojitosť – Vlastnosť dvoch pixelov so súradnicami (x, y) a $(\bar{x}, \bar{y})$. Dva pixely sú 8-spojité vtedy a len vtedy, ak platí $\max(|x - \bar{x}|, |y - \bar{y}|) = 1$. [5]



1.1-1 Vzťahy medzi 4-(8-) spojitosťou pixelov [5]

1.2Segmentácia obrazu

V počítačovom videní pojem segmentácia obrazu znamená proces rozdelenia digitálneho obrázku na viac častí (množín pixelov). Hlavným významom segmentácie je zjednodušiť, alebo zmeniť reprezentáciu obrázku na niečo také, čo by bolo možné ľahšie analyzovať [6]. Typicky sa používa na vyhľadávanie objektov na obrázkoch.

[1] V bežnom živote je takýto obrázok pomenovaný ako čierno-biely, no pri počítačových obrázkoch tento výraz používame pre obrázok, ktorý je zložený len z dvoch farieb (čiernej a bielej). Preto sa budeme držať týchto dvoch konvencií.

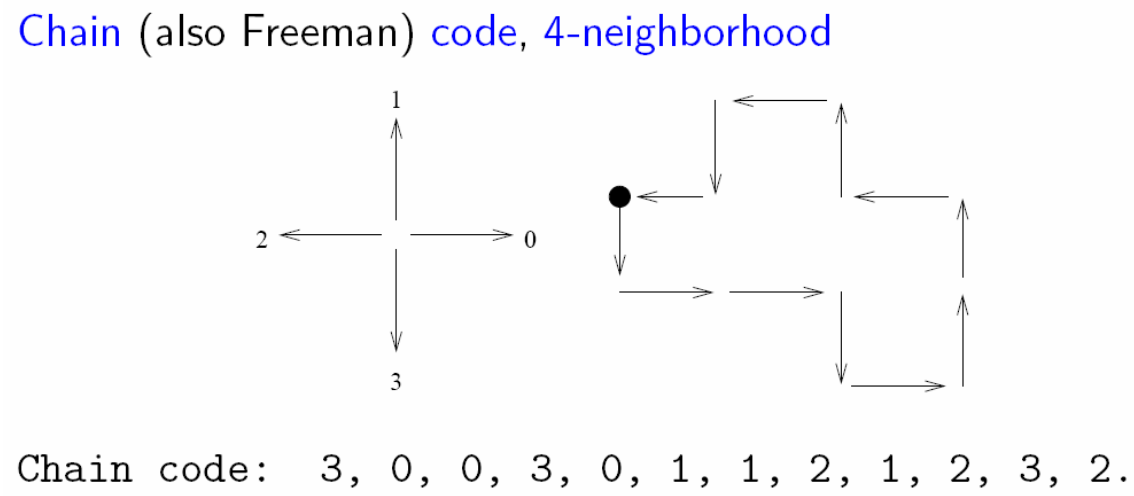
1.3Prahovanie – Thresholding

Prahovanie je najjednoduchšou metódou segmentácie obrazu. Ide o to, že z obrázku v odtieňoch šedej sa pomocou prahovania vyrobí binárny obrázok. Na začiatku sa určí nejaká prahová hodnota, podľa ktorej sa bude obrázok spracovávať. Potom sa všetky pixely s vyššou intenzitou ako zadaná prahová hodnota označujú ako pixely „objektu“. V opačnom prípade sa označia ako „pozadie“. Predpokladá sa, že objekt na obrázku je svetlejší ako pozadie. Toto sa nazýva prahovanie dohora (threshold above). Opakom toho je „prahovanie nadol“ (threshold below), pri ktorom sa za pixely patriace „objektu“ označujú také, ktoré majú intenzitu menšiu ako zadaná prahová hodnota. Ďalej sa používa výraz „vnútorné prahovanie“(threshold inside). V takom prípade sa na začiatku spracovávania určia dve prahové hodnoty. Za pixely patriace „objektu“ sa označujú tie, ktoré majú hodnoty ich intenzít, medzi týmito dvoma prahovými hodnotami. A nakoniec ešte spomenieme „vonkajšie prahovanie“ (threshold outside), ktoré je opakom vnútorného. Typicky sa pixely „objektu“ označujú číslom „1“ a pixely „pozadia“ číslom „0“. Z toho nám neskôr vznikne dvojfarebný obrázok, zložený len z bielej a čiernej. Existuje ešte adaptívne prahovanie. Jeho prahová hodnota sa určuje počas výpočtu z už prezretých pixelov. [7]

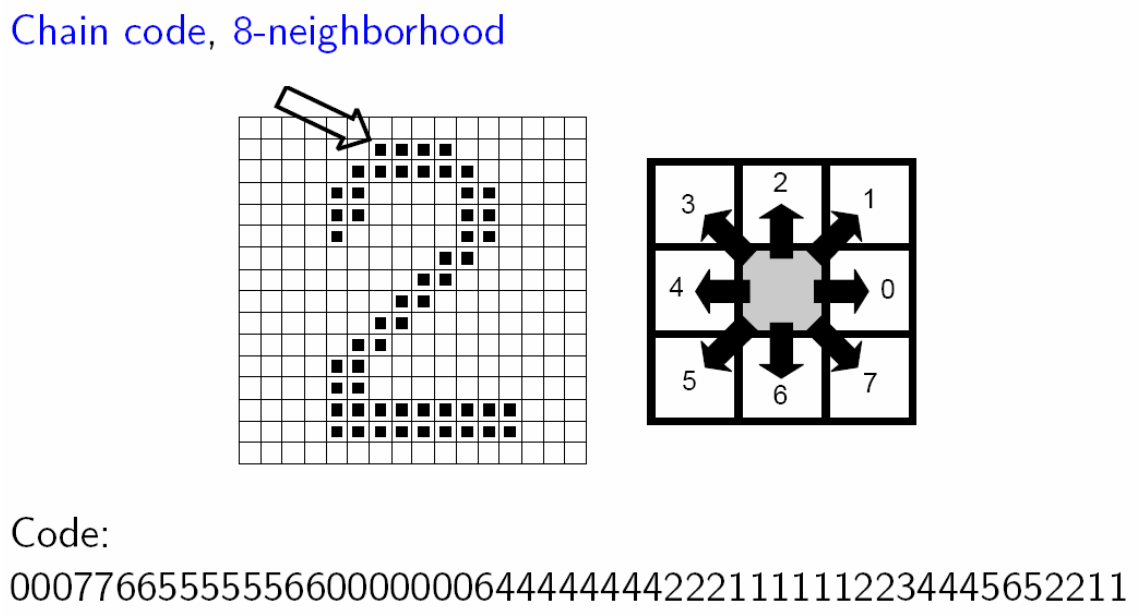
Hlavným účelom prahovania je zamaskovanie pixelov určitého rozsahu intenzít, alebo prevedenie obrázku v odtieňoch šedej na dvojfarebný čierno–biely obrázok. Dá sa použiť napríklad na extrakciu škvŕn s rovnakou svetelnou intenzitou.

1.4Retiazkový (Freemanov) kód

Slúži na zachytenie obrysov objektov na binárnych obrázkoch. Jedná sa o postupnosť čísel (smerov), ktoré postupne opisujú hranicu objektu na obrázku. Je možné použiť 4-smerovú, alebo 8-smerovú reprezentáciu, ako je znázornené na obrázkoch 1.3–1 a 1.3–2.



1.4-2 4–smerová reprezentácia Freemanovho kódu [8]



1.4-3 8–smerová reprezentácia Freemanovho kódu [8]

1.5Algoritmus vyhľadávania kontúr na obrázku

Algoritmy, ktoré sa venujú hľadaniu kontúr pracujú najčastejšie s binárnymi obrázkami. Použitím vzťahov 4-, alebo 8-spojitosti sú tieto obrázky rozložené do viacerých neprekrývajúcich sa 0-(1-) 4-spojitých alebo 8-spojitých komponent, z ktorých každá pozostáva z pixelov rovnakej hodnoty a to buď 0, alebo 1. Pre ktorékoľvek dva pixely z jednej komponenty môžeme medzi nimi nájsť cestu tvorenú pixelmi z tejto množiny, kde každé dva susedné majú vlastnosť 4- alebo 8-spojitosti. Medzi komponentami znázornenými na obrázku 1.5–1 môžeme nadefinovať ešte vzájomné vzťahy.

1.5-4Demonštrácia vzájomných vzťahov medzi komponentami [5]

1-komponenty W1, W2 a W3 sa nachádzajú vo vnútri obrázku (0-komponenta B1). Sú teda *priamo obklopené* komponentou B1.
0-komponenty B2 a B3 sú *vo vnútri* kompomenty W1.

1-komponenty W5 a W6 sa nachádzajú vo vnútri B4, ktorá je vo vnútri W3. To znamená, že tieto 1-komponenty sa nachádzajú *nepriamo* vo W3. Keďže ani jedna z týchto dvoch neobklopuje druhú povieme, že sú *na rovnakej hladine*.

Z doterajších pozorovaní môžeme povedať, že 0-komponenty sú doplnkami 1-komponent a oddeľujú ich od seba. Z toho dôvodu stačí uvažovať len 1-komponenty a študovať ich topologickú štruktúru. 0-pixely sa považujú za pozadie. 0-komponenta priamo obklpená 1-komponentou sa nazýva *diera* 1-komponenty. Za okrajový bod 1-komponenty môžeme označiť pixel, ktorý patrí do komponenty je 4-spojite spojený s 0-pixelom. Množina okrajových bodov sa nazýva hranica.

Každá 1-komponenta ma jednu vonkajšiu hranicu, ktorá ju oddeľuje od obklopujúcej 0-komponenty. Ďalej nula alebo viac dierových hraníc, ktoré oddeľujú 1-komponentu od obklopenej 0-komponenty. Zvyčajne nám teda vonkajšia hranica a dierové hranice udávajú celkový popis komponenty. Každá z hraníc komponenty sa nazýva kontúra. Jednotlivé kontúry sa reprezentujú podľa retiazkového (Freemanovho) kódu.

Existujú štyri varianty algoritmu na získanie popisov kontúr. Bližšie popísané v [9].

1. Prvý algoritmus nájde len extrémne vonkajšie kontúry na obrázku. Teda Podľa obrázku 1.5–1 sú to vonkajšie hranice oblastí W1, W2 a W3.
2. Druhý algoritmus nájde všetky vonkajšie kontúry. Na obrázku 1.5–1 je ich dokopy 8.
3. Tretí nájde všetky komponenty a vráti hierarchickú štruktúru, v ktorej je zachytený vonkajší okraj kontúry, a vnútorné okraje.
4. Štvrtý algoritmus vytvorí kompletný hierarchický strom kontúr. Napríklad na obrázku 1.5–1 je koreňom stromu oblasť B1. Synmi tohto vrcholu sú oblasti W1, W2 a W3. Pričom W1 a W2 už ďalších synov nemajú. W3 má jedného syna, ktorou je oblasť B4 a tá má zasa dvoch synov. Jej synmi sú W5 a W6.

Každý algoritmus prechádza jeden obrázok práve raz. Niekedy môže dôjsť aj k výnimke, keď je potrebné jednu kontúru prejsť viackrát. Algoritmy prechádzajú obrázky po riadkoch. V prípade, že algoritmus nájde nejaký bod, ktorý by mal patriť novému okraju, spustí sa procedúra, ktorá zachytí vonkajšiu hranicu retiazkovým kódom. Počas vykonávania tejto procedúry sa navštívené pixely označujú špeciálnymi pozitívnymi alebo negatívnymi hodnotami. Teda sa stráca informácia o pôvodnom vstupnom obrázku.

1.6Momenty

Momenty vo všeobecnosti sú akési vážené priemery intenzít pixelov na obrázku, alebo funkciami týchto momentov. Používajú sa na popis objektov po segmentácií obrázku. Najčastejšie vlastnosti objektov na obrázku, popísané momentmi, sú súradnice ťažiska objektu, alebo informácia o jeho orientácií. []

Pre spojitú funkciu dvoch premenných $f(x,y)$ je n-tý „hrubý“ moment [2] je pre $p,q = 1,2,3\ldots$, pričom $p+q=n$ definovaný ako

$$M_{pq} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x^p y^q f(x,y)\,dx\,dy$$

[2] Z aglického prekladu *„raw moment“*

Aplikovaním tohto na obrázok v odtieňoch šedej a pixelmi s intenzitami $I(x,y)$ dostávame hrubé momenty objektu M_{ij} nasledovne:

$$M_{ij} = \sum_x \sum_y x^i y^j I(x,y)$$

Tieto „hrubé“ momenty sa používajú hlavne pre výpočet súradníc ťažiska objektu. Okrem toho existujú aj tzv. „centrálne“ momenty, ktorých hlavným využitím je výpočet orientácií objektov. Bližšie informácie o momentoch je možné nájsť [10]

Súradnice (x,y) ťažiska objektu sa pomocou hrubých momentov vypočítajú:

$$x = \frac{M_{10}}{M_{00}},\ y = \frac{M_{01}}{M_{00}}.$$

Takto vypočítané ťažisko je veľmi presnou charakteristika objektu. Z geometrického hľadiska sa na toto riešenie dá pozerať ako hľadanie ťažiska bodov v rovine, pričom každému bodu pridelíme váhu podľa intenzity pixelu.

2Algoritmy na rozpoznávanie diód

Farby v počítači sa reprezentujú pomocou tzv. aditívneho RGB modelu [11]. To znamená, že každá sa dá poskladať z troch základných farieb – zložiek. Základnými farbami sú červená, zelená a modrá[3]. Jednotlivé farebné zložky sú reprezentované 8-bitovými číslami s rozsahom 0 až 255. Rôznymi kombináciami týchto zložiek môžeme poskladať až $255 \times 255 \times 255$ farieb. Čo je približne 16 miliónov. Na reprezentáciu farieb existujú ešte viaceré iné modely. Napríklad CMYK, CIE, HSV a HLS. [12]

Keďže používame diódy presne takých istých farieb, ktorých zložením sa reprezentuje farba v počítači, zavedieme pojem „dominantná zložka“ farby diódy. Tento pojem bude pre každú z diód reprezentovať farebnú zložku, u ktorej sa predpokladá, že na tejto dióde dominuje. Napríklad u červenej diódy to je červená zložka. Zavedieme taktiež pojem „recesívne zložky“ farby diódy, ktorým budeme na dióde nazývať ostatné dve zložky farby. Napríklad u červenej diódy to bude zelená a modrá.

Video sekvencia je zložená z viacerých obrázkov. Tieto sa postupne dostávajú do počítača a po jednom spracovávajú. Budeme preto často v texte hovoriť o jednotlivých obrázkoch. Každý pixel jednotlivých obrázkov je reprezentovaný pomocou spomínaného RGB modelu.

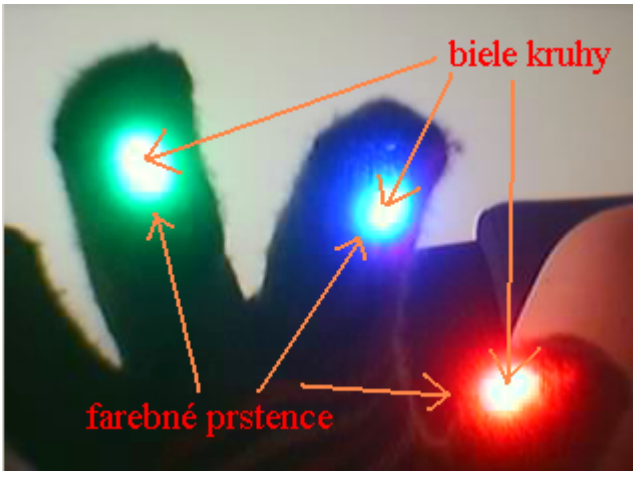
Algoritmy na rozpoznávanie diód by sme mohli podľa spôsobu vyhľadávania rozdeliť do dvoch skupín. Algoritmy založené na prezeraní jednotlivých pixelov a algoritmy založené na prezeraní kontúr. V prvom prípade sa snažíme pri pohľade na jeden pixel rozhodnúť o tom či patrí niektorej z diód alebo nie. V druhom prípade ide objekty, zložené z viacerých pixelov, o ktorých sa rozhodujeme, že či diódami sú alebo nie. Tento spôsob je síce oproti prvému menej efektívny (musíme všetky pixely obrázku prejsť viackrát) no diódy určí s obrovskou presnosťou.

V nasledujúcich podkapitolách budeme používať pojmy tmavšie a svetlejšie prostredie. Tmavším prostredím budeme označovať prostredie, v ktorom sa pri hľadaní kontúr nájde len veľmi málo svetelne výrazných objektov. Často tým budeme označovať šero, alebo tmu. Svetlejším prostredím budeme označovať prostredie, v ktorom sa pri hľadaní kontúr naopak nájde veľa výrazne svetelných objektov.

[3] V angličtine Red, Green, Blue.

Takýmto prostredím je napríklad miestnosť za jasného slnka, alebo osvetlená žiarovkou. Za výrazne svetelné objekty považujeme napríklad žiarovky, bielu stenu od ktorej sa svetlo odráža.

Z dôvodu, že diódy vyžarujú veľmi silné svetlo, je na obrázkoch jednotlivých diód zachytený biely kruh s prstencom danej farby diódy. Je to spôsobené preexponovaním [13] obrázku silným svetlom diódy. Snažíme sa preto pri práci vždy používať zníženú expozíciu kamery. Tým dosiahneme minimalizovanie veľkosti bielych kruhov v strede diód. Má to ešte výhodu aj v tom, že sa kamerou nezachytia svetelne málo výrazné objekty z pozadia, a tým sa nám výrazne uľahčí vyhľadávania diód.



1.6-5Vzhľad diód na jednom snímku

2.1 Maximalizácia dominantnej

Bol to prvý, veľmi naivný, pokus o získanie množín súradníc pixelov prislúchajúcich k jednotlivým diódam. K dióde sa z celého obrázku vybrali pixely, ktoré mali maximálnu hodnotu v dominantnej zložke na danom obrázku. Toto riešenie sa zavrhlo hneď, pretože sa do úvahy vôbec nebrali recesívne zložky. Za pixely patriace diódam sa teda okrem pixelov dominantnej farby, pokladali aj všetky biele pixely, ktorých RGB hodnoty sú 255,255,255, pretože ich dominantná farba je vo všetkých zložkách maximálna.

2.2Maximalizácia dominantnej mínus priemer recesívnych

V tomto prípade bola snaha o výber pixelov takých, ktoré majú maximálny rozdiel dominantnej zložky a priemeru recesívnych zložiek. Toto riešenie už bolo presnejšie. Napriek tomu sa ani toto riešenie neujalo. Do úvahy sa nebrala skutočnosť, že u diódy je potrebné mať hodnotu dominantnej zložky veľmi vysokú. Vysvetlíme si to na príklade. Predstavme si, že máme obrázok so zachytenou červenou diódou. V oblasti kde sa červená dióda nachádza, sa môže vyskytovať napríklad pixel s hodnotou 244 v červenej zložke a priemer v modrej a zelenej zložke môže byť 100. Teda rozdiel týchto dvoch čísel je 144. A dajme tomu, že 144 je maximum takto počítaných hodnôt na tomto obrázku. Potom by sa do výberu množiny pixelov dostal aj taký, ktorý má hodnotu červenej zložky 144 a priemer modrej a zelenej zložky 0. Ale takýto pixel sa len ťažko vyskytne v oblasti červenej diódy, pretože ako uvidíme neskôr hodnoty červenej zložky v oblasti červenej diódy sú určite nad 200.

2.3Dominantná 255, minimalizácia priemeru recesívnych

Táto metóda už zožala aj prvý úspech. Jednalo sa o vyberanie pixelov, ktoré majú dominantnú zložku rovnú 255 a minimálny priemer recesívnych zložiek. Do výsledného výberu sa pre konkrétnu diódu dostávalo len málo pixelov. Najčastejšie jeden a maximálne tri až štyri. Čo bolo výborné a všetky tieto pixely sa na obrázku vyskytovali v oblasti svojej diódy. Tomuto princípu veľmi pomáhalo zníženie expozície kamery na dosť malú hodnotu tak, aby boli viditeľné len diódy. Nebolo to však rezistentné voči externým zdrojom svetla v pozadí. Napríklad obyčajná stolová lampa namierená z pozadia na kameru. Táto vytvorí určite aspoň jeden pixel bielej farby. Pretože sme zatiaľ stále predpokladali, že sa diódy na obrázku vyskytujú, je lampa považovaná za niektorú z diód. Pri dostatočne tmavom pozadí a pri zníženej expozícií sa urobili jednoducho ťažiská vybraných pixelov. Tieto obiehali okolo prepáleného stredu po farebnom prstenci i keď sa dióda nepohybovala. Riešenie týchto nepresností bolo ponechané na spracovávač nájdených pixelov[4].

2.4Metóda hľadania kontúr

Budeme predpokladať, že pracujeme so zníženou expozíciou kamery na veľmi malú hodnotu. Je to kvôli minimalizácii svetla prichádzajúceho z okolitých objektov. Napríklad zo stien izby a podobne. Ďalej diódy vyžarujú svetlo veľmi silné, takže ich bude dobre vidieť a zníži sa tým aj ich biele presvecovanie snímkov z kamery.

Metóda využíva princíp prúdového spracovania dát. V prvom rade sa farebný obrázok prevedie obrázok v odtieňoch šedej. Diódy sú veľmi výrazné objekty z tohto dôvodu budú tomto obrázku vynikať bielou farbou. Z tohto sa odstráni šum pomocou vnútorného prahovania[5]. Toto spôsobí, že z objektov, ako napríklad diódy, vzniknú biele oblasti na čiernom pozadí. V tomto momente je potrebné tieto objekty identifikovať. Na to nám poslúžil prvý variant algoritmu na vyhľadanie kontúr na obrázku[6]. Pre takto získané kontúry sa na prevedenom čiernobielom obrázku (získanom z prvotného prevodu farebného na čiernobiely) podľa intenzít pixlov vypočítajú pomocou momentov ich presné ťažiská. Máme teda kontúry a ich ťažiská. Zostáva rozhodnúť o tom, ktorá kontúra prislúcha ku ktorej dióde. Najprv si ku každej kontúre zistíme súradnice obdĺžnika, ktorý ju obopína a začíname pracovať znovu z farebným obrázkom. V jednotlivých obdĺžnikoch si na farebnom obrázku zrátame priemerné hodnoty farieb v jednotlivých farebných zložkách. Nasleduje rozhodovanie o príslušnosti kontúr k diódam. Pre kontúru sa z vypočítaných hodnôt zistí najväčšia priemerná hodnota farby, v jednotlivých zložkách a priradí sa k dióde tejto farby. Výstupom z tejto časti sú súradnice ťažísk kontúr priradených k diódam. Tieto ťažiská veľmi presne určia polohy diód.

Táto metóda funguje priam ideálne, ale len ak sa nachádzame v tmavších priestoroch, alebo za šera, či úplnej tmy. V takomto prípade za prítomnosti diód sa tu objavujú presne tri kontúry príslušné k diódam. Za neprítomnosti diód sa nenájdu žiadne kontúry. Ak sa, ale nachádzame v jasnejších priestoroch, alebo máme v pozadí ešte iné zdroje svetla, keď nám na odstránenie svetelných obrazcov nepomôže ani zníženie expozície kamery, táto metóda úplne zlyháva. Zlyháva preto, lebo nemáme dostatočne silný mechanizmus na vyradenie kontúr, ktoré k diódam nepatria. Určite sa nestačí pozerať len ja jednu farebnú zložku kontúry.

[4] Myslí sa niektorý z algoritmov pre pohyb myši, ktorý dostane dáta od vyhľadávača pixelov.

[5] viď kapitola 1.3

[6] viď kapitola 1.5

Tento naivný spôsob priraďovania netreba hneď zatracovať. Môže nám vhodne poslúžiť ako heuristika. Aby sme mohli na prvý pohľad o kontúre povedať aspoň to, na akú farbu diódy je táto kontúra adeptom.

2.5Vytvorenie štatistík farebnosti diód

V predchádzajúcej kapitole sme preberali metódu, ktorá funguje dokonale v tmavšom prostredí za pomoci zníženej expozície kamery. Alebo v úplnej tme. Načrtli sme, že by bolo dobré vytvoriť nejaký mechanizmus, ktorý by nám v prípade lepších svetelných podmienok dokázal odfiltrovať kontúry neprislúchajúce oblastiam diód.

K účelu zozbierania dát týkajúcich sa farebnosti diód bola použitá metóda hľadania kontúr s naivným priraďovaním kontúr diódam. Pracovalo sa v tmavom prostredí. Bol napísaný program, ktorý pre každú kontúru prislúchajúcu k niektorej dióde zaznamenáva priemerné hodnoty, červenej, zelenej a modrej zložky farby. Ďalej tak isto ich pomery. Teda pomery červenej a zelenej, červenej a modrej, zelenej a modrej zložky farby a nakoniec pomer červenej, zelenej a modrej zložky farby dohromady. Takéto charakteristiky boli vybraté kvôli tomu, aby sa okrem farieb jednotlivých zložiek dali zachytiť aj závislosti, medzi farbami na jednej dióde. Toto meranie sa opakovalo 8700-krát. Neskôr sa za pomoci programu R [14] vytvorili stĺpcové grafy [15] týchto hodnôt. Tieto nám vizuálne načrtli správanie farieb na diódach.{pozri obrazovú prílohu.}. Nakoniec sa pomocou kvantilových funkcií [16] vypočítali intervaly, v ktorých sa vyskytuje 95% nameraných dát.

Zmerané štatistiky sa začali používať ako filtre kontúr vo svetlejších prostrediach. Výsledky už boli lepšie, ako v predchádzajúcom prípade. Ľahko sa vyraďovali kontúry nepatriace diódam. Na druhej strane nastávali aj výpadky. To znamená, že boli vyraďované kontúry patriace diódam. Ak si zoberieme, že slabšia kamera je schopná zachytiť 15 obrázkov za sekundu, tak nám výjde, že zhruba každých 7 sekúnd nastane pravdepodobne 5 výpadkov správneho určenia diód. Čo nám neskôr pri pohybe kurzorom myši spôsobí trhaný pohyb. Z toho dôvodu bol interval výskytu nameraných dát postupne zväčšený na 99,999%. Počet výpadkov sa znížil, na prijateľnú úroveň.

Nastáva nám otázka: Menia sa svetelné vlastnosti diód v odlišných prostrediach? Odpoveďou je samozrejme áno menia. Jednak síce vyžarujú vlastné svetlo, no pri zmene svetelných podmienok prostredia na ne dopadá viac či, menej svetla, a to mení ich svetelné vlastnosti. Aj tento faktor sa podpísal pod zvýšenie množstva výpadkov. Bolo teda potrebné upraviť výsledné intervaly, do ktorých by mali spadať farebné zložky diód a pomery týchto farieb. Intervaly, pre farby jednotlivých zložiek boli zväčšené doprava až na plnú hodnotu 255. Intervaly pomerov farieb boli podľa týchto úprav zmenené na želané hodnoty. Takto upravené intervaly farebnosti diód filtrovali neprislúchajúce kontúry ešte stále úspešne a veľmi sa minimalizovali výpadky. Za zníženej expozície to funguje veľmi dobre. Za automaticky, kamerou nastavovanej expozície dostatočne, pretože tu už nastáva viac výpadkov.

2.6Štatistická metóda

V predchádzajúcich dvoch kapitolách sme sa venovali metóde hľadania kontúr a vytvoreniu akéhosi štatistického filtra na vyraďovanie kontúr nepatriacich do oblasti diód. Týmto sme si pripravili pôdu, pre vysvetlenie jednej z najdôležitejších častí programu a tou je Štatistická metóda na hľadanie diód vo video sekvencii obrazu.

Základom tejto metódy je metóda hľadania kontúr s naivným priraďovaním kontúr k diódam. Naivné priraďovanie je v tomto prípade heuristikou, ktorá nám určí, na akú diódu je kontúra adeptom. Adept na niektorú z diód, sa neskôr otestuje, že či naozaj spĺňa podmienku príslušnosti, farebných zložiek a ich pomerov, do štatisticky

vypočítaných intervalov. V prípade, že adept na diódu uspeje, je k dióde priradený, a výstupom sú ťažiská všetkých diód, respektíve kontúr im prislúchajúcim.

2.7Metóda určovania farebnosti

Táto metóda je postavená znovu na vyhľadaní kontúr obrázku. Každá kontúra sa najprv heuristikou odhadne na akú diódu je adeptom. Potom sa spočíta konkrétna farebnosť (červenosť, modrosť, zelenosť), pre každý pixel v ohraničujúcom obdĺžniku kontúry, podľa vzorca:

$$Farebnosť = 2 \times dominantná\ zložka - súčet\ recesívnych$$

Z výsledných hodnôt farebnosti sa urobí priemer. Tento postup nám určuje akúsi priemernú farebnosť jednej kontúry. Napokon nastúpi rozhodovacie pravidlo o zamietnutí priradenia kontúry k dióde. Ak je farebnosť príliš malá priradenie sa zamietlo. Inak sa kontúra priradí k dióde.

Pri testovaní tento algoritmus fungoval zase len v tmavších prostredia s menším počtom kontúr. No na rozdiel od metódy s naivným prideľovaním mal mechanizmus na filtrovanie neprijateľných kontúr, a nemal žiadne výpadky. No nedal sa úplne využiť v dobre osvetlenom prostredí, z dôvodu nezamietnutia všetkých nesprávnych kontúr.

2.8Výsledná metóda použitá v programe

Vo výslednom programe bola na hľadanie diód najprv použitá iba samostatná metóda hľadania kontúr so štatistickým filtrovacím mechanizmom nadbytočných nájdených kontúr. Takto funguje, aj v jasnejšom, aj tmavšom prostredí. Nastáva malý počet výpadkov, ktorý je riešený pri rozpoznávaní gest. Z dôvodu úplnej minimalizácie výpadkov aspoň v tmavších prostrediach bol k metóde doplnený filtrovací mechanizmus s určovaním farebnosti.

3Gestá

Na obraze z kamery dostávame trojrozmerný svet v dvojrozmernom podaní. V tomto pohľade sú napríklad vzdialenosti medzi diódami relatívne. Pri väčšej vzdialenosti ruky od kamery sa nám môžu javiť, že sa nachádzajú bližšie pri sebe, ako pri menšej vzdialenosti ruky od kamery. Je teda potrebné brať na toto ohľad a snažiť sa vymyslieť gestá, ktoré nie sú závislé na určovaní vzdialeností.

Ak chceme úplne nahradiť myš je nutné mať určené gestá pre všetky udalosti, ktoré môže generovať. Znamená to mať pripravené gestá pre pohyb myši, stlačenie a pustenie každého z tlačidiel. V neposlednom rade aj gestá pre skrolovanie kolieskom myši hore a dole. To znamená, že ich potrebujeme mať celkom 6 plus gestá pre pohyb.

Pre pohyb myši určite zrejme nemusíme vymýšľať nejaké špeciálne. Stačí ak budeme sledovať, ako sa na obrázku premiestňujú diódy. Zrejme je možné určovať pohyb kurzora myši sledovaním troch, dvoch, alebo jednej diódy. Prvé dva spôsoby by bolo možné využiť v prípade, že nevieme s dostatočnou presnosťou určiť polohu diód. Tretí zase v prípade, že sme schopný diódy identifikovať s veľkou presnosťou.

3.1Prvý nápad

Úplne na začiatku vývoja bola myšlienka gest založená na sledovaní všetkých troch diód. Kurzorom sa malo pohybovať sledovaním ich spoločného ťažiska. Kliknutie ľavým tlačidlom myši bolo naplánované spojením palca a ukazováka. Pre pravé tlačidlo

zasa spojením palca a prostredníka. Skrolovať bolo v pláne otáčaním diód v smere a protismere hodinových ručičiek.

Gesto bolo použité v začiatkoch vývoja len pre testovanie pohybu myši. Implementácia ďalších spomínaných vlastnosti by zahrnovala zapamätavávanie si akejsi histórie pohybu. Toto by mohlo zvýšiť dobu odozvy po vykonaní gesta. Napríklad pri kliknutí myšou by sme museli zistiť, že sa momentálne nejedná o pohybovanie myši, ale o pohyb prstov smerom k sebe.

3.1-6Klik ľavým tlačidlom myši

3.1-7 Skrolovanie

3.2„Boľavý palec“

Ešte v čase, keď sa nedali presne určovať diódy, bolo vymyslená metóda „boľavý palec“. Kvôli pohybu myši sa sledovala modra, a zelená dióda. Kurzorom sa pohybovalo podľa ťažiska oboch. Kliknutie ľavým tlačidlom myši sa prevádzalo premiestnením palca červenej diódy za vertikálnu hranicu modrej a zelenej diódy. Po dlhšom čase používania z toho človeka bolel palec. Riešenie bolo nepraktické aj z toho dôvodu, že sa tam už nedali vymyslieť gestá na skrolovanie.

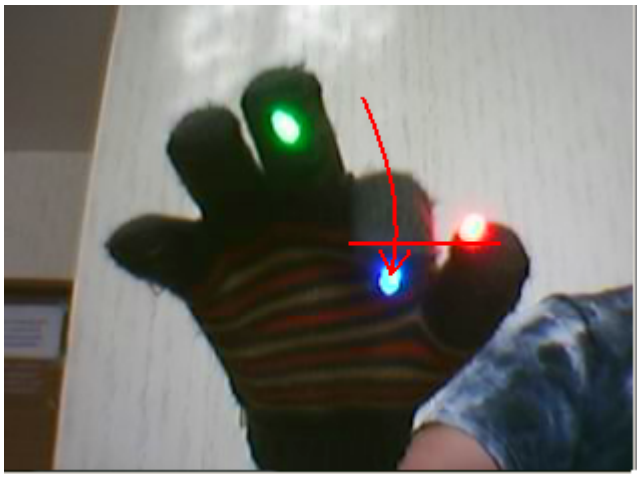
Metóda sa používala na odladenie algoritmov klikania a pohybu myši. Z dôvodu nerozšíriteľnosti na zastúpenie všetkých funkcií myši a bolesti palca sa gesto zavrhlo.

3.2-8Klik ľavého tlačidla myši

3.3Finálne gestá

V štádiu vývoja aplikácie, keď sa polohy diód dali veľmi presne určiť sa pohyb kurzoru myši začal reprezentovať sledovaním jednej diódy. (Konkrétne červenej, umiestnenej na palci.) Ruka je v základnej polohe tak, ako keď užívateľ drží v ruke myš. Gesto pre klikanie myšou bolo navrhnuté tak, že pre kliknutie ľavého tlačidla sa modrá dióda dostane pod úroveň horizontálnej roviny, v ktorej sa nachádza červená dióda. Obdobne pre kliknutie pravého tlačidla. Akurát sa kontroluje zelená dióda. K týmto gestám bolo potrebné pridať ešte skrolovanie myši. Preto sa zaviedlo gesto pre prepnutie do skrolovacieho režimu. A na skrolovanie hore a dole sa používajú gestá ako na kliknutie ľavým a pravým tlačidlom myši.

3.3-9Základná poloha – finálne gesto



3.3-10Kliknutie ľavým tlačidlom myši – finálne gesto

3.3-11 Kliknutie pravým tlačidlom myši – finálne gesto

3.3-12 Prepnutie do skrolovacieho režimu – finálne gesto

4Rozpoznávanie gest

Pri rozpoznávaní gest sa nám stačí zamerať len na ťažiská diód a ich polohu na jednotlivých obrázkoch. V takomto prípade, je rozpoznanie gesta veľmi jednoduché. Pre ťažiská červenej, zelenej a modrej diódy budeme používať názvy červené, zelené

a modré ťažisko. Ďalej kliknutie ľavého resp. pravého tlačidla myši budeme nazývať výrazom ľavý resp. pravý klik.

4.1Pravý a ľavý klik

Vysvetlíme si len kliknutie ľavým tlačidlom myši. Kliknutie pravým tlačidlom myši sa urobí obdobne, akurát namiesto modrého ťažiska budeme sledovať zelené.

Sledujeme modré a červené ťažiská. V momente keď y-nová súradnica modrého, je ostro menšia ako y-nová súradnica červeného ťažiska. Nastane a odošle sa udalosť stlačenie ľavého tlačidla. Následne ak bude y-nová súradnica modrého, ostro väčšia ako y-nová súradnica červeného ťažiska odošle sa udalosť pustenia ľavého tlačidla myši.

4.2Skrolovanie

V tomto prípade predpokladáme, že x-ová súradnica zeleného je ostro menšia ako x-ová súradnica modrého ťažiska. To plynie z umiestnenia diód na prostredníku a ukazováku a prípadné prekríženie prstov sa neberie do úvahy. Z tohto dôvodu nám stačí sledovať polohy červeného a zeleného ťažiska. Ak x-ová súradnica červeného bude ostro menšia ako x-ová súradnica zeleného ťažiska aplikácia sa prepne do režimu skrolovania. Ak sa potom červené ťažisko vráti naspať do základnej polohy, je možné gestami pre pravý a ľavý klik skrolovať. Po následnom opakovaní sa aplikácia vráti do pôvodného režimu.

4.3Pohyb myši

Pri pohybe kurzorom myši sa pamätajú súradnice červeného ťažiska na predchádzajúcom obrázku. Podľa súčasnej polohy sa vypočíta o koľko, a ktorým smerom sa má pohnúť. Tým, akým spôsobom sa kurzorom myši pohybuje, sa budeme venovať v nasledujúcej kapitole.

5Algoritmy pre pohyb kurzorom myši a klikanie

Doteraz sme sa zaoberali problematikou identifikácie diód a rozoznávaním gest. Je určite jasné, že tam naša cesta nekončí. Je potrebné sa ešte pozrieť na to ako doterajšie poznatky použiť pre dosiahnutie verného nahradenia myši. V jednotlivých častiach sa budeme venovať riešeniu problémov s týmto spojených.

5.1Klikanie

Na jednoduchom kliknutí zrejme nie je nič zaujímavé. Na jedno kliknutie je potrebné odoslať dve udalosti. A to stlačenie tlačidla a pustenie tlačidla.

Zaujímavé to začína byť ak chceme vykonať dvojklik alebo viacklik. Žiadny užívateľ by, pri vykonaní dvoch kliknutí, nedokázal udržať ruku bez pohnutia v rovnakej polohe hneď na prvý krát. Po prvom kliknutí by sa mohlo stať, že sa kurzor o malý úsek posunie a teda vo výsledku to nebude dvojklik, ale dva jednoduché kliky. Preto bolo nutné tento problém vyriešiť. Riešenie spočívalo v tom, že sa po odoslaní, už prvej udalosti (stlačenie tlačidla), zablokuje pohyb kurzoru. Najprv bolo navrhnuté zablokovanie na nejaký čas. Za blokovací časový úsek sa bral v systéme windows nastavovateľný „Double Click Time“ [17]. No tento úsek môže byť dlhý aj 500 milisekúnd a je nemožné užívateľa vždy pri každej udalosti stlačenia tlačidla blokovať v pohybe kurzorom myši na taký dlhý čas. Stratili by sme na interaktivite. Preto toto riešenie bolo doplnené ešte potrebou prekonania určitej vzdialenosti. Pričom sa po dobu časového limitu stále počíta vzdialenosť, ktorú užívateľ zatiaľ prekonal. Ak je táto dohodnutá vzdialenosť prekonaná. Pohyb kurzorom sa odblokuje.

Celkové riešenie teda spočíva v tom, že pri odoslaní každej udalosti stlačenia tlačidla myši sa pohyb zablokuje dovtedy, kým užívateľ neprekoná stanovenú vzdialenosť, alebo neuplynie časový limit nastavovateľný v systéme. To má za následok umožnenie užívateľovi bez väčších problémov vykonávať dvoj a viac kliky myšou bez straty akejkoľvek interaktivity s užívateľom.

5.2Pohyb kurzoru myši v systéme windows

Určite si čitateľ už všimol, že ak drží v ruke myš a pohybuje ňou veľmi pomaly, kurzor sa pohybuje veľmi presne a značne pomaly. V prípade, že myšou pohne rýchlejšie, jej kurzor prejde väčšiu dráhu na obrazovke, ale s menšou presnosťou. Pri pomalom pohybe myšou, ňou potrebujeme prejsť dlhší úsek na to, aby sme prešli celú šírku obrazovky. Pri rýchlom pohybe nám stačí par centimetrov.

Stručne si vysvetlíme základný princíp na akom funguje pohybovanie kurzorom myši v systéme windows. Pre presnejšie informácie o tejto oblasti môže čitateľ nájsť [18].

Windows dostane z ovládača [19] myši údaje o jej aktuálnom pohybe. Zistí okamžitú rýchlosť myši. Z tejto rýchlosti podľa špeciálnych funkcií, ktoré sa podobajú viacerým prepojeným lineárnym priamkam, vypočíta okamžitú rýchlosť kurzora na obrazovke. Podľa tejto rýchlosti myšou pohne.

5.3Vyhladzovanie pohybu

Na začiatku nebolo možné určiť diódy veľmi presne. Aj keď rukou nebolo pohybované dostávali sme polohy diód značne odlišné. Čo v konečnom dôsledku spôsobovalo veľké trhanie kurzorom. Pristúpilo sa teda k pamätaniu histórie zaznamenaných pohybov rukavice. Základnou myšlienkou bolo, že ak máme uloženú už nejakú históriu pohybu a z nej spočítanú hodnotu ťažiska kurzoru. Tak táto hodnota by mala mať pri ďalšom pohnutí váhu veľkosti histórie. Nová informácia o premiestnení diódy by mala mať váhu jeden. Výsledná aktuálna poloha kurzoru sa vypočítala váženým priemerom [20] ťažiska histórie pohybu a novej polohy kurzoru.

Pohyb myši bol málo interaktívny. Zmena polohy kurzoru sa prejavovala s časovou odozvou úmernou dĺžky pamätanej histórie pohybu. Aj kvôli tomuto sa pristúpilo k hľadaniu algoritmov, ktoré by boli schopné hľadať diódy s väčšou presnosťou.

5.4Lineárny pohyb

Najjednoduchšie riešenie pohybu. Vezme sa stará a nová poloha diódy. Vypočíta sa smerový vektor [21] pohybu. Tento sa vynásobí konštantou a kurzorom sa posune v smere tohto vektora.

Riešenie nie je veľmi vhodné, pretože pri nastavení konštanty príliš vysoko sa kurzor pohybuje po veľkých skokoch a tým znemožňuje užívateľovi prácu na malých úsekoch obrazovky. Pri nastavení konštanty príliš malej nie sme schopný ani pri prejdení celej šírky obrázku prejsť kurzorom celú šírku obrazovky.

5.5Upravený lineárny pohyb

Nápadom bolo použiť pri lineárnom pohybe na násobenie smerového vektora dve konštanty. Jednu väčšiu a jednu menšiu. Aby bolo umožnené, pri malej zmene pohybu, pohnúť sa o menší kus, pri veľkej zmene, o väčší. Merala sa veľkosť smerového vektora pohybu. V prípade, že jeho veľkosť nepresiahla dohodnutú hranicu tak sa násobil menšou konštantou. V opačnom prípade väčšou.

V konečnom dôsledku to znamenalo, priblíženie sa skutočnému pohybu myšou vo windowse. Pohyb bol lepší ale stále to nemalo žiadaný efekt kvôli, skákavému pohybu kurzora bo častiach a viditeľnej zmene pohybu kurzora myši, kedy bolo možné vizuálne spozorovať, že sa kurzor začal pohybovať oveľa rýchlejšie.

5.6Dve lineárne priamky a prahovanie pohybu

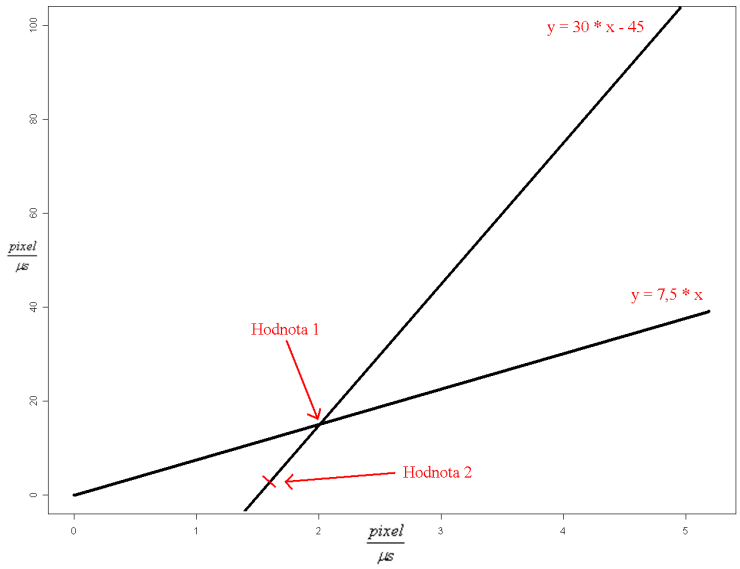
V doterajších riešeniach pohybu sa nebral vôbec do úvahy čas. Pohyb bol závislý na „frame rate“[7]. Čo by pri výkonnejších, alebo slabších počítačoch, či kamerách nemuselo dobre fungovať. Začala sa teda merať okamžitá rýchlosť diódy. Veľkosť smerového vektora, medzi posledným a aktuálnym snímkom z kamery, sa vydelila časom, ktorý ubehol od posledného snímku. Táto rýchlosť sa potom prepočítala na okamžitú rýchlosť kurzora na obrazovke. Z tejto rýchlosti sa vypočítala dráha, ktorú by tento kurzor za daný čas mal ubehnúť. Touto dráhou sa vynásobil normalizovaný smerový vektor pohybu diódy na obrázku.

Zostáva nám vysvetliť ako sa prepočítavala okamžitá rýchlosť diódy na okamžitú rýchlosť kurzora na obrazovke. Najprv boli určené dve priamky s rovnicami $y = 7{,}5 \times x$ a $y = 30 \times x - 45$. Rýchlosti kurzora sa vypočítavali dosadzovaním do jednej alebo druhej rovnice. V prípade, že rýchlosť diódy bola nižšia ako vopred stanovená hranica (na obrázku je označená ako Hodnota 1), na výpočet sa použila prvá rovnica. V prípade, že rýchlosť diódy presiahla spomínanú hranicu tak sa na výpočet používala druhá rovnica. Podľa tejto rovnice sa počítalo dovtedy kým rýchlosť diódy neklesla pod druhú stanovenú hranicu (na obrázku je označená ako Hodnota 2). V takom prípade sa začalo počítať opäť podľa prvej rovnice. Prvá hraničná hodnota bola nastavená ako x-ová súradnica priesečníku priamok v rovine (mala hodnotu 2). A druhá sa zvolila tak aby mala menšiu hodnotu ako prvá (mála hodnotu 1,6). To z dôvodu aby, ak užívateľ pohne rukou rýchlo, a potom chce následne zastaviť, tak aby bolo toto spomalenie výraznejšie a tým sa stal pohyb kurzorom myši presnejší.

Priamky sa vypočítali tak, že sa sledovali rýchlosti diód na jednotlivých obrázkoch pri lineárnom pohybe a odhadlo sa, kde by mala byť hranica, keď je potrebné rýchlosť kurzora zvýšiť. Týmto spôsobom sa určila prvá hraničná hodnota a odhadlo sa, že pri takejto rýchlosti by mal mať kurzor rýchlosť 15 $pixel / \mu s^{-1}$. Ďalej priamka musí navyše prechádzať počiatkom súradnicovej sústavy. Pomocou týchto dvoch bodov sme dostali prvú priamku. Druhá musela teda prechádzať bodom $[2;2]$ a podľa grafu sa znovu odhadlo aby prechádzala bodom $[1{,}5;0]$. Takýmto spôsobom sme teda dostali aj druhú rovnicu priamky.

Našou ďalšou snahou bolo umožniť užívateľovi nastavovanie rôznej senzitivity pohybu kurzora po obrazovke. Nastavovanie dvoch lineárnych priamok by bolo dosť náročné. Preto sa realizoval nápad vytvoriť parabolu. Mali sme k dispozícii už dva body ($[0;0]$ a $[2;15]$), ktorými by sme mohli parabolu viesť. Bolo potrebné ešte určiť tretí bod. Za tretí bod sa zvolil niektorý vysoko položený bod na druhej priamke. Z týchto troch bodov sa už dala parabola vypočítať.

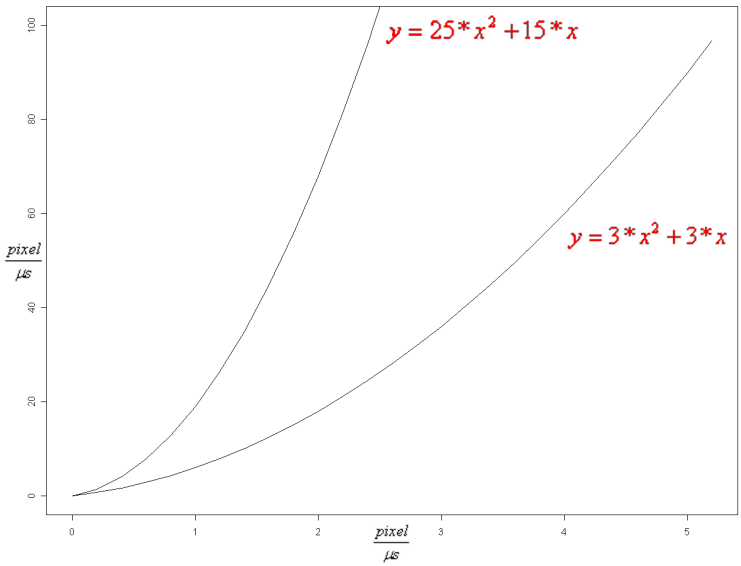
[7] Frekvencia akou kamera produkuje jednotlivé snímky



5.6-13 Graf dvoch lineárnych priamok na prepočet rýchlosti

5.7Parabolické prepočítavanie rýchlosti

Algoritmus pre pohybovanie myšou je v podstate rovnaký ako algoritmus dvoch lineárnych priamok. Líši sa len v prepočítavaní okamžitej rýchlosti kurzora myši. Prepočítavanie sa robí za pomoci parabolickej rovnice $y = ax^2 + bx$, kde a a b sú premenné nastavované užívateľom. Premenná a sa môže pohybovať v intervale $\langle 3,25 \rangle$ a b v intervale $\langle 3,15 \rangle$. Parabola bola najprv vypočítaná staticky z viacerých bodov dvoch lineárnych priamok opísaných v predošlej kapitole. Hranice premenných boli neskôr vizuálne odladené nastavovaním v programe, tak aby bol pohyb pekný a plynulý. Na nasledujúcom obrázku pomocou grafu zobrazené rozpätie možných nastavení rýchlosti pohybu. $y = 25 * x^2 + 15 * x$



5.7-14 Graf nastaviteľnosti senzitivity myši

6Užívateľská príručka

6.1Potrebné nástroje

Pre používanie programu Minority Report je potrebné mať rukavicu s tromi rôznofarebnými vysokosvietivými diódami a webkameru podporujúcu vlastné nastavenie expozície.

6.1.1Rukavica s tromi rôznofarebnými diódami

Jedná sa o rukavicu, ktorá má na palci upevnenú červenú, na ukazováku modrú a na prostredníku zelenú diódu. Tieto sú paralelne zapojené na 9V baterku, umiestnenú na vrchnej časti ruky.

6.1-15. Rukavica s rôznofarebnými diódami

Príprava diód

Na každú diódu je potrebné pripevniť odpor 35 Ω a vodiče potrebnej dĺžky, tak aby dosiahli k zdroju napätia. (9V baterku)

Zhotovenie rukavice

Pripravené diódy s vodičmi a odpormi, vpletieme do rukavice (alebo inak umiestníme) tak, aby diódy boli umiestnené zdola na končekoch prstov a konce vodičov vyústili na hornej strane ruky. Tieto sa paralelne pripoja na zdroj napätia, ktorý je najlepšie všiť do rukavice, aby ho nebolo vidieť.

6.1-16. Vpletenie vodičov do rukavice a paralelne spojenie

6.1-17. Vpletenie vodičov do rukavice pohľad zdola

6.1.2Webkamera

Program k svojmu behu potrebuje kameru, ktorá umožňuje nastaviť expozíciu. Teda kamera musí takúto funkciu podporovať. Inak sa pri pohybe myši môže vyskytnúť veľmi trhavý pohyb kurzora myši, až nefunkčnosť. S kamerou, ktorá nepodporuje vlastné nastavenie expozície treba pracovať pri minimálnom osvetlení miestnosti. V takom prípade je možné aj zvýšiť hodnotu parametra *„Down Threshold*[8] *“*.

6.2Inštalácia programu

O kompletné nainštalovanie programu sa postará inštalátor, ktorý všetky potrebné súčasti nahrá, do *„Program Files/Minority Report“* a vytvorí zástupcu na pracovnej ploche. Po nainštalovaní je možné program okamžite používať.

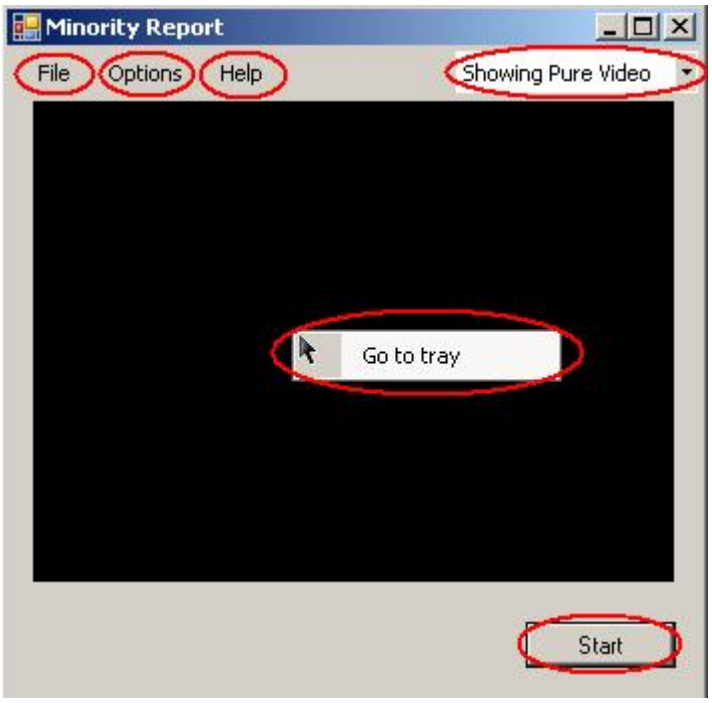
6.3Používanie aplikácie

Program je možné používať v pracovnom prostredí windows a ovládať pomocou neho kurzor myši, za použitia webkamery, predpísanej rukavice a správnych gest rukou.

[8] viď kapitolu o nastavovaní parametrov programu

6.3.1Základný popis aplikácie

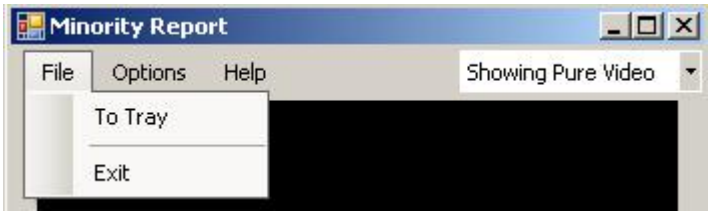
Po spustení aplikácie sa objaví okno aplikácie, v ktorom je hore umiestnené menu so štyrmi položkami: *„File“*, *„Options“* *„Help“* a v pravo hore je umiestnený combobox pre výber konkrétneho behu aplikácie. V Strede okna je čierny obdĺžnik, v ktorom sa bude zobrazovať video z kamery a v pravom dolnom rohu okna je tlačidlo *„Start“*, ktoré aplikáciu spustí. Po kliknutí pravým tlačidlom myši do aplikačného okna sa zobrazí ponuka *„Go to tray“*, ktorá túto aplikáciu skryje do Tray-u.



6.3-18Okno aplikácie

Položka menu „File“

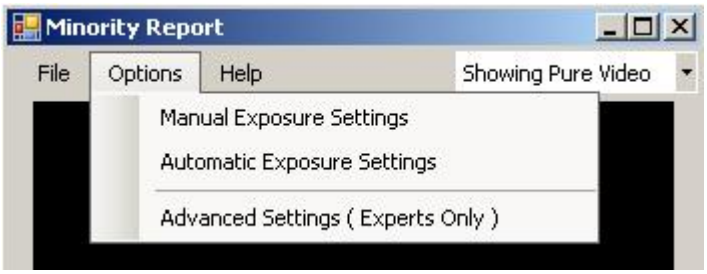
Po kliknutí na túto položku sa zobrazí podmenu s dvoma možnosťami: „To Tray“, ktorá skryje aplikáciu do Tray-u, a položka „Exit“, ktorá ukončí beh aplikácie a zatvorí okno aplikácie.



6.3-19. Popis Položky v Menu "File"

Položka menu „Options“

Po kliknutí na túto položku sa zobrazí podmenu s troma možnosťami: *„Manual Exposure Settings“*, *„Automatic Exposure Settings“*, *„Advanced Settings“*.



6.3-20. Popis Položky v Menu "Options"

Manual Exposure Settings

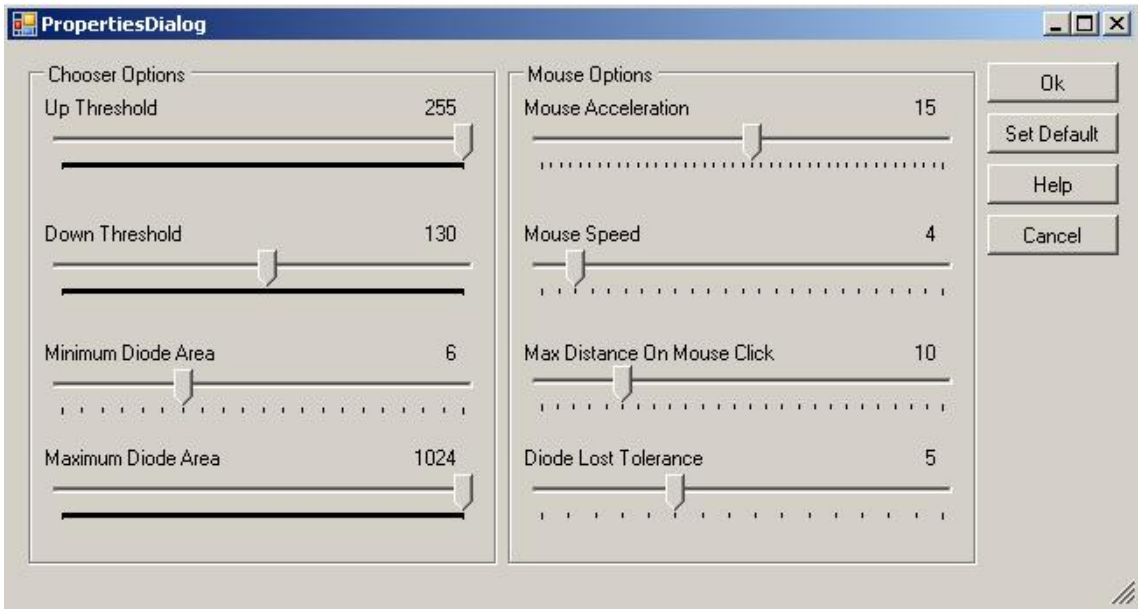
Po kliknutí na túto možnosť sa zobrazí okno, ktoré umožní nastavovať vlastnú expozíciu kamery. Toto okno je vlastným oknom ovládača kamery, ktorý je dodávaný výrobcom zariadenia.

Automatic Exposure Settings

Táto možnosť umožňuje explicitne nastaviť pre program vhodnú expozíciu. Funguje v prípade, že je v comboboxe vybratá možnosť *„Only mark diods“* alebo *„Do mouse actions“*. Program ju sám nastaví.

Advanced Settings

Po kliknutí na túto položku sa zobrazí dialógové okno, v ktorom sa dajú dolaďovať interné nastavenia programu. Tieto nastavenia sa dajú rozdeliť do dvoch skupín: *„Chooser Options“* a *„Mouse Options“*. Prvé uvedené nastavenia sa týkajú rozoznávania diód na obraze z kamery. A druhé uvedené sa týkajú nastavení myši. Ak je pustené video v niektorých z režimov *„Only mark diods“* alebo *Chooser Options“* tak sa zmeny ktoréhokoľvek z parametrov prejavujú okamžite.



6.3-21. Properties Dialog

Up Threshold, Down Threshold

Tieto nastavenia sa používajú pri prvotnom spracovaní jedného obrázku z video sekvencie, kde sa vyradia všetky intenzity pixlov, ktoré sú menšie ako Down Threshold

a väčšie ako „Up Threshold“. Tieto pixely sa nahradia čiernymi pixelmi. Preto je nutné aby Up Threshold bol ostro väčší ako „Down Threshold“. Pre neskúsených užívateľov a správny beh programu je doporučené mať „Up Threshold“ nastavený na hodnotu 255, a „Down Threshold“ sa môže meniť v závislosti na intenzite diód a okolitého svetla. Ak je priveľa okolitého svetla je dobre „Down Threshold“ zvýšiť. Ak sú diódy už príliš slabé, je dobré tento parameter znížiť.

Minimum Diode Area, Maximum Diode Area

Tieto dve nastavenia určujú aký minimálny, či maximálny obsah môžu mať nájdené diódy. Tieto nastavenia sa neodporúča meniť. Je to možné upraviť v prípade ak sa nachádzame príliš ďaleko od kamery. V takom prípade je možnosť zvýšiť hodnotu parametra *„Minimum Diode Area“*

Mouse Acceleration, Mouse Speed

Tieto nastavenia určujú rýchlosť a zrýchlenie kurzora myši pri jeho ovládaní rukavicou. Užívateľ ich môže podľa potreby meniť tak aby sa mu s programom dobre pracovalo.

Max Distance On Mouse Click, Diode Lost Tolerance

Po vykonaní gesta nahradzujúceho klik tlačidlom myši sa na istý čas kurzor prestane hýbať a v pozadí sa počíta kam by sa za ten čas presunul. Ak tento úsek presiahne parameter *„Max Distance on Mouse click“*, tak sa kurzorom opäť bude dať pohybovať.

Pri vyhľadávaní diód sa niekedy stane, že program na obrázku nenájde diódu, aj keď sa tam nachádza. V tom prípade sa počká ešte určitý čas bez toho, aby sa po znovu nájdení diódy kurzor reinicializoval. Tento čas sa nastavuje parametrom *„Diode Lost Tolerance“*.

Položka menu „Help“

V tejto sekcii môžete nájsť informácie o programe, jeho verzii a informácie o používaní programu.

Popis možností výberu v Comboboxe

Na výber sú tri možnosti: *„Showing Pure Video“*, *„Only Mark Diodes“*, *„Mouse Actions“*. Ak je vybraná prvá spomínaná možnosť, tak po kliknutí na tlačidlo start sa spustí zobrazovanie iba čistého videa z kamery. Po výbere druhej možnosti sa bude zobrazovať video so zníženou expozíciou. Vo videu sa budú vyhľadávať diódy a označovať ich poloha. V prípade, že bude vybraná tretia možnosť, tak okrem zobrazovania videa a označovania diód, bude možné pomocou rukavice ovládať kurzor myši.

6.4Gestá pre ovládanie kurzora myši

Po spustení aplikácie, vybraní položky *„Mouse Actions“* v comboboxe a následnom kliknutí na tlačidlo start, je možné ovládať kurzor myši za pomoci rukavice snímanej webkamerou a to vykonávaním nasledujúcich gest rukou.

6.4.1Základná poloha ruky

Ruku mame otočenú dlaňou k objektívu kamery, pričom prsty, ukazovák a prostredník, sú v takej polohe aby sa na výslednom videu, modrá a zelená dióda, nachádzali nad horizontálnou hranicou, ktorá je medzi nimi a červenou diódou.

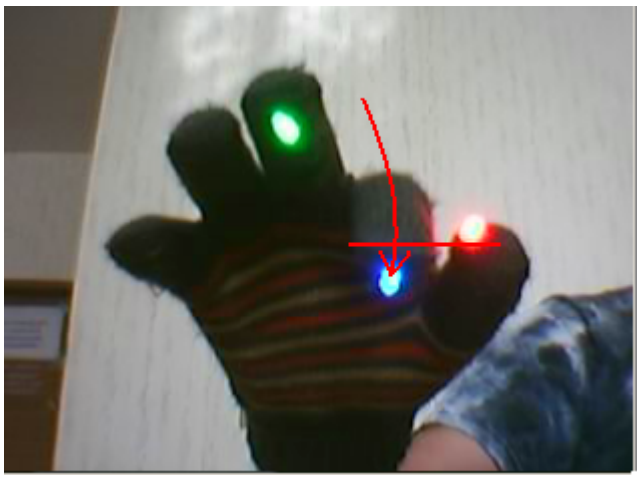
6.4-22 Základná poloha

6.4.2Pohyb kurzora myši

Program sleduje pohyb červenej diódy na videu z kamery. Podľa toho, ako človek pohne rukou, sa pohne aj kurzor myši na obrazovke.

6.4.3Klikanie

V prípade, že sa modrá dióda dostane pod úroveň červenej, nastane klik ľavého tlačidla myši. Ak sa pod úroveň červenej diódy dostane zelená, nastane klik pravého tlačidla myši.



6.4-23 Gesto kliknutia ľavým tlačidlom myši

6.4-24 Gesto kliknutia pravým tlačidlom myši

6.4.4Skrolovanie dokumentu

V prípade potreby je možné, gestom ruky, aplikáciu prepnúť do režimu skrolovania, a to tak, že červená dióda sa horizontálne presunie za vertikálnu úroveň zelenej diódy, a vráti sa do základnej polohy. V tomto režime je možné gestom pre kliknutie ľavým tlačidlom myši skrolovať nadol, a gestom kliknutia pravým tlačidlom skrolovať nahor. Po ukončení skrolovania je potrebné sa vrátiť opäť do režimu klikania, použitím toho istého gesta, ktoré sme použili pre zapnutie režimu skrolovania.

6.4-25 Zapnutie a vypnutie režimu skrolovania

6.5Riešenie problémov

6.5.1ProxyTrans.ax – chyba po vybratí kamery

Pri spustení programu a následnom výbere kamery, sa môže vyskytnúť chyba nasledujúceho znenia:
„ProxyTrans.ax could not be located, register it using RegSRVR32.exe“. ProxyTrans je filter z DirectShow a je potrebný pre používanie kamery. Tento filter je registrovaný automaticky, aj pri inštalácii programu Minority Report. Ale môže sa stať, že táto registrácia z nejakých dôvodov neprebehne úspešne. V takomto prípade je potrebné spustiť program *„Program Files/Minority Report/RegisterProxy.exe“*, ktorý spomínaný problém vyrieši a program bude možné opäť normálne používať.

6.5.2Kurzor myši sa pohybuje trhane

Je možné, že v rukavici je už slabý zdroj napätia. Vymeňte zdroj napätia, alebo v *„Options/Advanced Settings“* znížte hodnotu nastavenia *„Down Threshold“* na nižšiu tak, aby program označoval diódy za každých okolností do bielych rámčekov.

Je zapojený úplne nový zdroj napätia a problém sa neodstránil. V takomto prípade je najlepšie nesvietiť priamo do kamery žiadnou s diód, alebo v *„Options/Advanced Settings“* zvýšiť hodnotu nastavenia *„Down Threshold“* na vyššiu.

7Programátorská dokumentácia

Celá aplikácia sa skladá z dvoch častí : Ovládacia časť a Aplikačná časť. Ovládacia časť je napísaná v jazyku C#, za pomoci knižnice WinForms. Aplikačná časť je napísaná v jazyku C/C++ a pripojená ako .dll knižnica k ovládacej časti.

7.1Aplikačná časť

Dá sa rozdeliť do viacerých častí: OpenCV časť, Logika Aplikácie a Prístupové funkcie k Aplikácii.

7.1.1OpenCV

K behu aplikácie využívame knižnicu OpenCV, ktorá sa postará o veci technického charakteru, ako je napríklad: výber kamery, zobrazovanie videa do okna a bezpečné ukončenie používania kamery.

K spracovávaniu obrazu používame modul cvcam, ktorý poskytuje štandardné funkcie potrebné pre inicializáciu kamery, ako napríklad zadefinovanie callback funkcie, ktorá je volaná na každý frame[9], z video streamu[10], ďalej výber kamery, nastavenie veľkosti renderovaného videa. Modul cvcam štandardne nepodporuje programové nastavovanie expozície. Keďže zdrojové kódy sú verejne prístupné, tak bola táto funkcionalita dopísaná. A tým bol modul cvcam rozšírený o funkcionalitu nastavovania expozície pre operačný systém windows a používania DirectShow.[22]

Rozšírenie cvcam

Z potreby nastavovanie expozície kamery, bol modul cvcam rozšírený o ďalšiu funkcionalitu opísanú nižšie. Snahou bolo dodržiavať konvencie modulu cvcam pre názvoslovie funkcií a návratové hodnoty.

```
int cvcamActualCameraEnabled();
```

Funkcia vráti číslo aktuálne používanej kamery.

```
int cvcamGetProperty2(int camera, const char* property, void* value);
```

V module cvcam existuje funkcia s názvom cvcamGetProperty s rovnakými typmi parametrov aj návratovou hodnotou. Aby nebolo zasiahnuté do zdrojového kódu modulu cvcam. Bola implementovaná táto funkcia, ktorá reaguje len na nami definované property a vráti špeciálne hodnoty opísané nižšie.

```
int cvcamSetProperty2(int camera, const char* property, void* value);
```

[9] V programatorskej dokumentácii budeme využívať tento anglický názov pre jeden snímok zachytený vo video sekvencií obrazu.

[10] V programátorskej dokumentácii budeme využívať názvu video stream namiesto video sekvencia obrazu.

V cvcam existuje funkcia s názvom cvcamSetProperty. Bola preto implementovaná podobná metóda, ktorá nastavuje nami definované property.

```
struct cvcamExposureRanges
```

Pridali sme definíciu štruktúry, ktorá v sebe obsahuje hodnoty potrebné pre zistenie aktuálnych nastavení expozície. Táto štruktúra obsahuje päť premenných typu long. „Min“ „Max“ – reprezentujú minimálnu a maximálnu možnú expozíciu, ktorú dovoľuje kamera nastaviť. „Step“ – reprezentuje minimálny možný krok, o ktorý je možno expozíciu kamery zmeniť. „Default“ reprezentuje, defaultnu hodnotu nastavenia expozície. „Flags“ môže nadobúdať dve hodnotu definované v DirectShow a to sú CameraControl_Flags_Auto = 0x0001, ktorý znamená že expozícia je automatická, alebo CameraControl_Flags_Manual = 0x0002, expozícia je nastavovaná manuálne. Používa sa ako výstupná hodnota funkcie cvcamGetProperty2 pre property CVCAM_EXPOSURE_RANGES.

```
struct cvcamExposureValues
```

Táto štruktúra obsahuje dve premenné typu long. Premenná „Value“ reprezentuje aktuálnu hodnotu nastavenej expozície. Premenná „Flag“ obsahuje hodnotu buď CameraControl_Flags_Auto alebo CameraControl_Flags_Manual. Táto štruktúra sa používa ako vstupný parameter funkcie cvcamSetProperty2 a ako výstupný parameter funkcie cvcamGetProperty2 v prípade, že meno property je CVCAM_EXPOSURE_CURRENT_VALUE opísané nižšie.

```
const char CVCAM_EXPOSURE_RANGES[] = "current exposure settings";
```

Nami definovaný názov prečítateľnej vlastnosti, ktorý môže byť vstupným parametrom funkcie cvcamGetProperty2. V tomto prípade je návratovou hodnotou štruktúra s názvom cvcamExposureRanges.

```
const char CVCAM_EXPOSURE_CURRENT_VALUE[] = "current exposure values";
```

Mnou definovaný názov prečítateľnej aj nastaviteľnej vlastnosti, ktorý môže byť vstupom cvcamGetProperty2 aj cvcamSetProperty2. Návratovou hodnotou je v tomto prípade štruktúra cvcamExposureValues.

Prístupové funkcie k aplikácii

Jedná sa o funkcie, ktorých hlavičky sú definované v súbore ExportFunctions.h. Pomocou týchto funkcií je možné do aplikačnej časti pristupovať z ovládacej časti. Popíšeme si ich jednotlivú funkcionalitu a význam pre beh aplikácie. Na začiatku súboru je `#define MINORITY_EXPORT EXTERN_C __declspec(dllexport)`, ktorý vyexportuje hlavičky, funkcii vonku z dll súboru. Pred každou z nasledujúcich funkcii je napísané `MINORITY_EXPORT`.

```
int run(void * handle)
```

Funkcia, zobrazí dialóg pre výber kamery, inicializuje cvcam modul, nastaví príslušné parametre kamery a spustí zobrazovanie videa. Ako parameter dostáva handle [23] na okno, do ktorého sa bude zachytené video zobrazovať. Návratová hodnota môže byť 0 – inicializácia prebehla v poriadku, -1 – ak sa nepodarilo nainicializovať cvcam modul a -2 ak je zobrazený dialóg pre výber kamery a niekto sa pokúsi túto funkciu spustiť ešte raz.

```
int stopWorking()
```

Funkcia ukončí prácu s modulom cvcam, teda ukončí spracovávanie a zobrazovanie videa, a odalokuje všetky používané objekty.

```
void setInternalProperty(nSettings::Properties prop, double value)
```

Umožňuje nastaviť jednu z vlastností nastaviteľných v PropertiesDialógu. Na nejakú hodnotu. Nie je ošetrovaný interval rozsahu hodnôt.

```
void setExposure(bool manualy)
```

Umožní nastavovať expozíciu. V prípade, že parameter *„manualy“* má hodnotu *true*, zobrazí okno umožňujúce expozíciu nastaviť manuálne, inak spustí programové nastavenie expozície, na hodnotu potrebnú pre chod aplikácie.

```
void setCallBack(Callback call)
```

Nastaví jeden z definovaných možností behu aplikácie. Možnosti behu aplikácie sú definované v enume Callback. Užívateľ si niektorú z možností môže vybrať v comboboxe, v pravom hornom rohu aplikačného okna.

```
int ScrollingFlag()
```

Táto funkcia zisťuje či pri behu aplikácie je aplikácia v režime skrolovania, alebo klikania. Návratová hodnota je -1 – v prípade, že aplikácia beží s iným callbackom ako *„MOUSE_ACTIONS“*, 0 – v prípade, že je nastavený režim klikania, 1 – v prípade, že je nastavený režim skrolovania.

Logika Aplikácie

V tejto časti za zameriame viac na algoritmy použité pri rozpoznávaní diód a samotný pohyb myši, ako na opis funkcií. Detaelnejší popis funkcií, objektov a metód je možné si pozrieť v dokumentácii generovanej programom Doxygen [24].

Celá aplikácia je rozdelená do dvoch častí. Prvá časť sa stará o správnu identifikáciu jednotlivých diód na jednotlivých framoch z video streamu, nájdenie reprezentatívnych bodov a vykreslenie bielych obdĺžnikov okolo nájdených diód. Druhá časť pracuje už len na dátach, ktoré dostane od prvej časti. Stará sa o rozpoznávanie gest ruky a implementuje aj algoritmy pohybu kurzorom myši, klikania a skrolovania.

V aplikácii sa nachádza aj trieda Settings. Je implementovaná ako singleton [25]. Túto na začiatku nainicializuje ovládacia časť programu. Z tejto triedy si potom obe časti berú potrebné nastavenia parametrov.

Rozpoznávanie diód

Diódy rozpoznáva trieda, ChooserWithStats, ktorá implementuje interface Chooser. Rozpoznávanie sa deje vo virtuálnej metóde choose_pixels. Ako parameter dostáva aktuálny frame, a tzv. Container, do ktorého sa uloží výsledok. Aktuálny frame, sa najprv konvertuje na obrázok v odtieňoch šedej. Z obrázku sa potom odstránia nepotrebné pixely tak, že sa zavolá funkcia cvThreshold. Táto funkcia prebehne dvakrát. Prvý krát zmení všetky pixely s intenzitou menšou ako parameter DOWN_THRESHOLD na pixely čiernej farby. Druhý krát zmení pixely s intenzitou väčšou ako parameter UP_THRESHOLD na pixely s čiernou farbou a ostatné zmení na pixely z bielou farbou. Na takto pozmenený obrázok sa aplikuje vstavaná OpenCV funkcia cvFindContours [5], ktorá pomocou algoritmu na vyhľadávanie kontúr vyberie všetky výrazné objekty. V každom ohraničujúcom obdĺžniku kontúry sa spočíta priemer vo všetkých farebných zložkách v pôvodnom farebnom obrázku a ťažisko kontúry pre

ďalšie spracovanie. Nasleduje rozhodovanie, či daná kontúra, je alebo nie je jednou z diód. Na určovanie je možné použiť dve rôzne metódy:

Štatistická metóda

Získané priemery sa porovnávajú voči dopredu pripraveným a spočítaným štatistikám diód. Tieto boli spočítané mimo programu. Jedná sa o akési hranice, ktoré musia takto získané priemery splňovať. Štatistiky hovoria o pomeroch priemerov jednotlivých zložiek. Boli zozbierané za úplnej tmy určovaním diód len pomocou toho, že dióda je objekt s najvyšším priemerom farby v jednej zložke. Táto štatistická metóda za predpokladu, že sa na obrázku dióda nachádza, na 99,999% o nej rozhodne správne. Za predpokladu, že sa tam dióda nenachádza, a sú v pozadí nejaké výrazné objekty napríklad zdroj bieleho svetla, táto metóda rozhodne vo veľa prípadoch správne. I napriek tomu, že táto metóda pracuje správne až na 99,999% niekedy nastávajú výpadky (diódy sa nenájdu). Tieto sa riešia v časti pre pohyb myši a klikanie.

Metóda určovania farebnosti

Táto metóda funguje na 100% v tmavších prostrediach, kde správne identifikuje diódy, ale reaguje aj na iné farebné veci v pozadí. Preto túto metódu používame len, pri malom počte nájdených kontúr. U každej kontúry sa počíta farebnosť. Pre každý pixel sa vypočíta rozdiel farba1-(farba2 + farba3). Z týchto hodnôt sa urobí priemer a ak tento priemer presiahne nejakú danú hranicu môžeme o danej dióde povedať, že podľa farebnosti spĺňa podmienku. Nepočítajú sa rozdiely pre každú farbu, na začiatku prebehne nižšie opísaná heuristika, ktorá rozhodne o nejakej kontúre na akú farbu je adeptom.

Pre každú kontúru sa určí, na akú farbu je táto kontúra adeptom. Toto sa určuje podľa maximálneho priemeru vo farebných zložkách. Potom sa podľa počtu nájdených kontúr rozhodne, ktorá z metód sa použije na otestovanie diód. Metóda buď potvrdí alebo vyvráti, že daná kontúra je alebo nie je diódou.

Niekedy sa môže stať, že na jednej dióde vzniknú dve kontúry, ktoré splňujú podmienky. V tomto prípade sa pristúpi k zlučovaniu kontúr, no testovanie už neprebieha.

Po takomto spracovaní a rozhodnutí o diódach nastáva fáza kreslenia obdĺžnikov ohraničujúcich diódy, a nastavenie ťažísk diód do kontainera pre ďalšie spracovanie.

Rozpoznávanie gest ruky

Gestá rozpoznáva trieda MouseMoverMarker, ktorá implementuje interface Marker. Rozpoznávanie sa deje vo virtuálnej metóde mark_pixels, ktorá ako parameter dostáva aktuálny frame, a Container, v ktorom sú vybraté ťažiská diód. Na začiatku sa prekontroluje scrollingFlag, že či ho netreba zapnúť, alebo vypnúť. Potom sa vykonajú kliky myšou (v prípade zapnutého režimu skrolovania sa vykoná buď skrol hore alebo dole). Ďalej sa skontroluje, že či náhodou nebolo kliknuté. Ak áno tak sa čaká kým kurzor myši prejde určitú vzdialenosť alebo kým neuplynie systémový doubleClickTime. Ešte sa za pomoci triedy velocitySolver vyráta okamžitá rýchlosť kurzora vo video streame (okamžitá rýchlosť červenej diódy). Napokon sa preráta okamžitá rýchlosť diódy na okamžitú rýchlosť kurzora na obrazovke pomocou parabolickej funkcie. Potom sa zráta dráha, ktorú má na ploche kurzor prejsť a touto dráhou sa vynásobí normalizovaný jednotkový vektor pohybu diódy v streame.

Klikanie myšou

Skontrolujú sa pozície ťažísk modrej a zelenej diódy voči ťažisku červenej diódy. Ak je y-nová súradnica niektorého z nich menšia ako y-nová súradnica odošle sa udalosť mouse button down (Buď ľavý alebo pravý). Ak v ďalších framoch nastane táto udalosť znovu tak sa ignoruje dovtedy kým nenastane opačná udalosť, ktorou je mouse button click up. Teda od tejto chvíle sa pozeráme na to či sa y-nová súradnica konkrétneho tlačidla nezväčšila nad hodnotu y-novej súradnice červenej diódy.

Čakanie po kliku

V prípade kliku je možné, že bude nasledovať ďalší klik. Preto sa začne v každom ďalšom frame počítať smerový vektor pohybu červenej diódy a meria sa jeho aktuálna veľkosť. Vektor sa počíta pripočítavaním nového k starým hodnotám. Zároveň sa však spustí aj časovač. Zastavenie pohybu kurzora po kliku tlačidlom sa ukončí v prípade, že veľkosť počítaného vektora presiahne nastavovateľný parameter, alebo uplynie systémový doubleClickTime.

Výpočet okamžitej rýchlosti diódy

Na tento účel slúži trieda velocitySolver, v ktorej beží interný časovač a po každom frame sa počíta čas, ktorý uplynul od posledného framu. Trieda velocitySolver dostane dráhu, ktorú prekonala dióda, od posledného výskytu a vráti jej okamžitú rýchlosť.

Výpočet okamžitej rýchlosti kurzora na obrazovke

Na tento výpočet sa používa parametrizovateľná kvadratická funkcia. Koeficientom pri kvadratickom člene je hodnota nastaviteľného parametra MOUSE_ACCELERATION. Pri lineárnom člene je MOUSE_SPEED. Absolútny člen je nulový.

7.2Ovládacia časť

V prostredí Visual Studia bolo nadizajnované jednoduché okienko. Do stredu okna bol umiestnený PictureBox, do ktorého bude dll-ková knižnica zobrazovať video stream. (Handler na toto okno sa predáva ako parameter vo funkcii run). Funkcie, ktoré sa používajú na komunikáciu s dll-kovou knižnicou, boli logicky oddelené od ostatných a sú kompilované ako statické funkcie triedy DllCals. Za zmienku snáď stojí ešte nastavovanie parametrov aplikácie z *„Properties dialógu“*. Hlavnou myšlienkou bolo umožniť užívateľom za behu meniť tieto parametre a tak okamžite vidieť výsledok nastavenia bez toho, aby museli kliknúť na tlačidlo ok. Z toho dôvodu je potrebné si pamätať staré hodnoty aby sa v prípade nesúhlasu s nastavením, toto mohlo vrátiť do pôvodného stavu. Hodnoty sa pamätajú v *„Properties dialógu“* ako privátne premenné.

7.3Kompilácia

Ku kompilácii je potrebné mať nainštalované knižnice OpenCV a správne si ich nalinkovať k projektu. Štandardne inštalujú na rovnakú úroveň ako inštalačný adresár Visual Studia (do Program Files). Je potrebné najprv prekompilovať modul cvcam s priloženými vlastnými zdrojovými kódmi. Ku kompilácii modulu cvcam je potrebné mať nainštalovaný DirectX SDK a ešte „Extras“ knižnice s DirectShow. Po úspešnom zvládnutí prekompilovania cvcam. A nalikovaní knižníc OpenCV by už s kompiláciou nemal byť žiadny problém. Je potrebné Aplikáciu skompilovať v Release móde, a preklad optimalizovať na rýchlosť. V debug verzii sa môže stať, že obraz bude trhaný.

Na priloženom CD je uložený celý projekt generovaný Visual Studiom 2008 aj zdrojové kódy. Po nakopírovaní projektu na lokálny disk stačí pre jeho kompiláciu najprv nainštalovať knižnice OpenCV do *„Program Files“* (v prípade, že je v tomto adresári nachádza aj priečinok Visual Studia). Potom je potrebné nahradiť zdrojové kódy modulu cvcam (OpenCV/otherlibs/cvcam), so zdrojovými kódmi modulu cvcam na priloženom CD. Ďalej je potrebné nahradiť cvcam.lib (OpenCV/lib) a cvcam100.dll (OpenCV/bin) predpripravenými súbormi toho istého názvu nachádzajúcimi sa na priloženom CD[11]. Po vykonaní týchto krokov by už teda projekt mal byť dobre nastavený a pripravený na skompilovanie. Výhodou tejto druhej varianty je, že nie je potrebné kompilovať Extras knižnice s Directshow kvôli opätovnému prekompilovaniu cvcam modulu.

[11] viď „Obsah priloženého CD“

Záver

V predloženej sme dôkladne prebrali algoritmy, z ktorých sa postupne vyvinuli algoritmy používané na rozpoznanie diód vo video sekvencií. Ďalej sme sa postupným vývojom dopracovali k algoritmu pohybu kurzorom myši, používajúceho parabolickú funkciu, ktorá prepočítava okamžitú rýchlosť diódy, na okamžitú rýchlosť kurzora na obrazovke. Po vhodnej úprave by sa dal aj integrovať do systému windows, ako prepočítavač rýchlosti skutočnej myši. Vyvinuli sme aplikáciu, ktorá je schopná nahradiť jedno počítačové vsutpné zariadenie a to počítačovú myš. Aplikácia je nasadená v operačnom systéme windows, umožňujúca pohybovanie kurzorom myši a zastúpenie funkcií tlačidiel.

Zaujímavou myšlienkou do budúcnosti, by bolo pokúsiť sa nahradiť nejakým spôsobom klávesnicu, aby bola úplne dovŕšená myšlienka ovládania počítača bez použitia klávesnice a myši. Možno by bolo vhodné navrhnúť a implementovať celý operačný systém založený na tejto myšlienke.

Použitá Literatúra

1. Jan De Bont, Minority Report, Sci-Fi Thriller, 2002
2. Intel Corporation, OpenCV, http://www.intel.com/technology/computing/opencv/index.htm, knižnica v jazyku C/C++
3. D.H. Ballard, C.M. Brown; Computer Vision, Prentice-Hall Inc New Jersey, 1982, ISBN 0-13-165316-4
4. Wikipedia – The Free Encyclopedia, Obrázok v odtieňoch šedej, http://en.wikipedia.org/wiki/Grayscale, webová stránka, (Júl 2008)
5. Intel Corporation, OpenCV Reference manual, http://www.cs.unc.edu/Research/stc/FAQs/OpenCV/OpenCVReferenceManual.pdf, Manuál ku knižnici OpenCV, (Júl 2008)
6. Wikipedia – The Free Encyclopedia, Segmentácia obrazu, http://en.wikipedia.org/wiki/Segmentation_(image_processing), (Júl 2008)
7. Wikipedia – The Free Encyclopedia, Prahovanie, http://en.wikipedia.org/wiki/Thresholding_(image_processing), webová stránka, (Júl 2008)
8. prof. Václav Hlaváč, Retiazkový kód, prezentácia k prednáške http://cmp.felk.cvut.cz/~hlavac/TeachPresEn/15ImageAnalysis/51-2DImageDescritpiton.pdf, (Júl 2008)
9. S. Suzuki, K. Abe., Topological Structural Analysis of Digital BinaryImages by Border Following. CVGIP, v.30, n.1. 1985, pp. 32-46.
10. Wikipedia – The Free Encyclopedia, Momenty obrázku, http://en.wikipedia.org/wiki/Image_moments, webová stránka, (Júl 2008)
11. Wikipedia – The Free Encyclopedia, RGB model, http://en.wikipedia.org/wiki/RGB, webová stránka, (Júl 2008)
12. Wikipedia – The Free Encyclopedia, Farebné modely, http://en.wikipedia.org/wiki/Color_model, webová stránka, (Júl 2008)
13. Wikipedia – The Free Encyclopedia, Expozícia, http://en.wikipedia.org/wiki/Exposure_(photography), webová stránka, (Júl 2008)
14. R. Gentleman, R. Ihaka, R-project, Program na štatistické výpočty, http://www.r-project.org, (Júl 2008)
15. Wikipedia – The Free Encyclopedia, Stĺpcový diagram, http://en.wikipedia.org/wiki/Bar_chart, webová stránka, (Júl 2008)
16. Wikipedia – The Free Encyclopedia, Kvantilová funkcia, http://en.wikipedia.org/wiki/Quantile_function, webová stránka, (Júl 2008)
17. Microsoft msdn Library, funkcia getDoubleClickTime, http://msdn.microsoft.com/en-us/library/ms646258(VS.85).aspx, webová stránka, (Júl 2008)
18. Microsoft, Balistika kurzora myši, http://www.microsoft.com/whdc/archive/pointer-bal.mspx, webová stránka, (Júl 2008)
19. Wikipedia – The Free Encyclopedia, Ovládač zariadnia, http://en.wikipedia.org/wiki/Device_driver, webová stránka, (Júl 2008)

20. Wikipedia – The Free Encyclopedia, Vážený priemer, http://en.wikipedia.org/wiki/Weighted_mean, webová stránka, (Júl 2008)
21. Wikipedia – The Free Encyclopedia, Smerový vektor, http://en.wikipedia.org/wiki/Direction_vector, webová stránka, (Júl 2008)
22. Microsoft msdn Library, DirectShow, Knižnica podporujúca multimediálne prehrávanie, http://msdn.microsoft.com/en-us/library/ms783323(VS.85).aspx, (Júl 2008)
23. Wikipedia – The Free Encyclopedia, Handle, http://en.wikipedia.org/wiki/Handle_(computing), webová stránka, (Júl 2008)
24. Dimitri van Heesch, Doxygen, Program na tvorbu automatickej dokumentácie, http://www.stack.nl/~dimitri/doxygen/
25. Wikipedia – The Free Encyclopedia, Návrhový vzor Syngleton, http://en.wikipedia.org/wiki/Singleton_pattern, webová stránka, (Júl 2008)
26. Linda Shapiro and George Stockman, *Computer Vision*, Prentice Hall, 2001. ISBN 0-13-030796-3

Obsah priloženého CD

Priečinok „Bc práca“

V tomto priečinku sa nachádza text tejto práce uložený vo formáte pdf.

Priečinok „Doxygen“

V tomto priečinku sa nachádza dokumentácia k projektu vygenerovaná programom doxygen. Prehliadanie je možné začať spustením súboru „index.html“

Priečinok „Inštalácia“

Tu sa nachádzajú inštalačné súbory programu „Minority Report“ použiteľného na ovládanie kurzoru myši pomocou rukavice. Inštaláciu je možné spustiť súborom „setup.exe“.

Priečinok „Video“

V tomto priečinku môžete nájsť demonštračné videa, k programu a preberaným metódam použitým na vyhľadávanie diód. Video „aplikácia.wmv“ obsahuje jednoduchý popis ovládania aplikácie. Videa „maxdom.wmv“, „dominantna255.wmv“, „domMinusRec.wmv“ a „metKont.wmv“ ukazujú, ako fungujú preberané metódy 2.1, 2.2, 2.3 a 2.4. Ďalej „thresholding.wmv“ ukazuje ako funguje algoritmus prahovania obrázku preberaného v 1.3. A napokon v ukážkach „solitare.wmv“ a „miny.wmv“ si je možné pozrieť, ako sa dajú za pomoci vytvoreného programu a rukavice hrať najznámejšie windowsové hry „Solitare“ a „Hľadanie mín“.

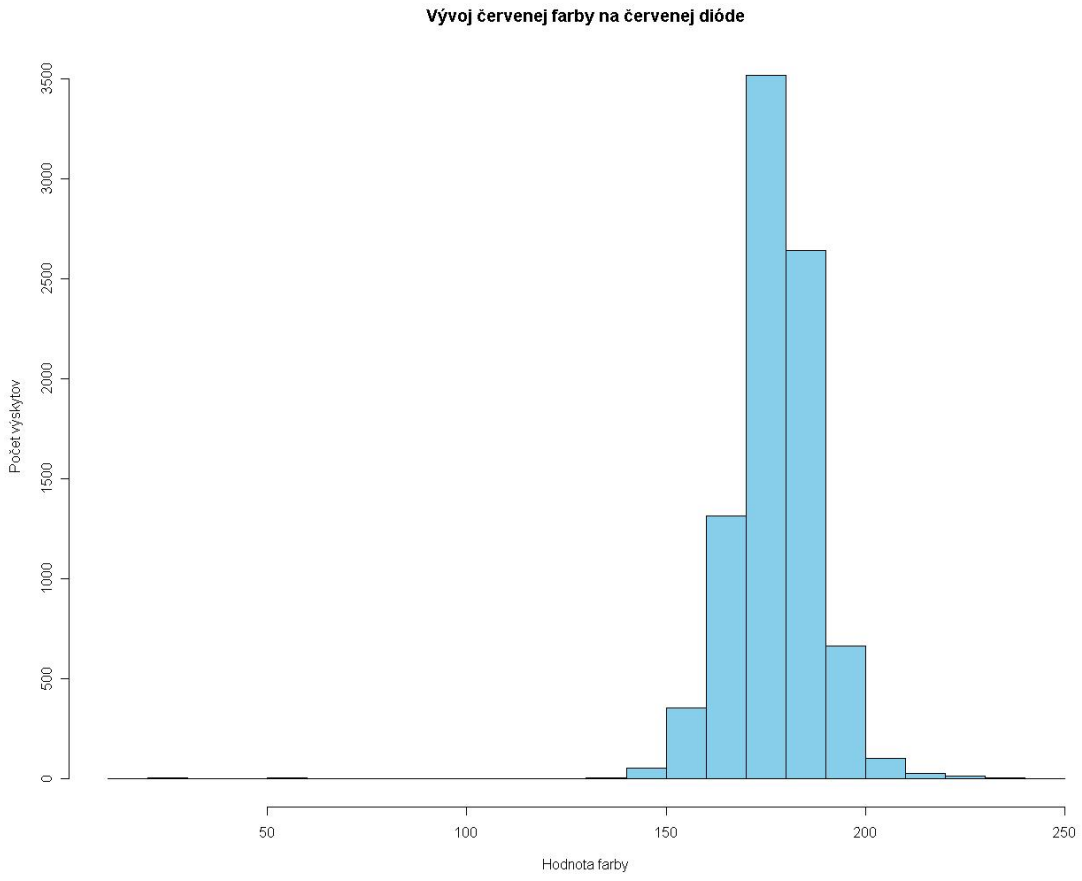
Priečinok „Projekt“

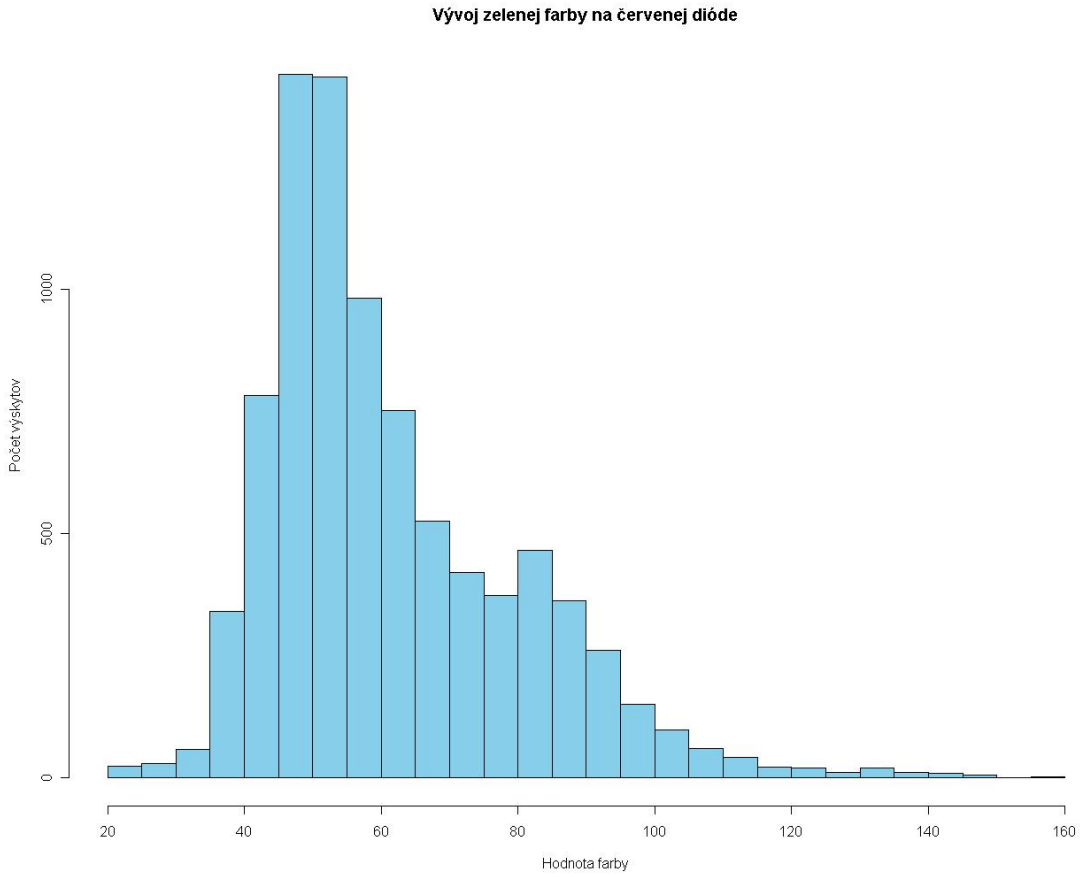
V priečinku projekt je možné nájsť dva adresáre a to „MinorityReport“ a „cvcamModul“. V prvom menovanom sa nachádza solution, k projektu s názvom „MinorityReport.sln“. V druhom sa nachádzajú upravené zdrojové kódy cvcam, cvcam.lib a cvcam100.dll, ktoré sú potrebné pre skompilovanie projektu MinorityReport. Hovorili sme o nich v Programátorskej dokumentácii, v časti kompilácia.

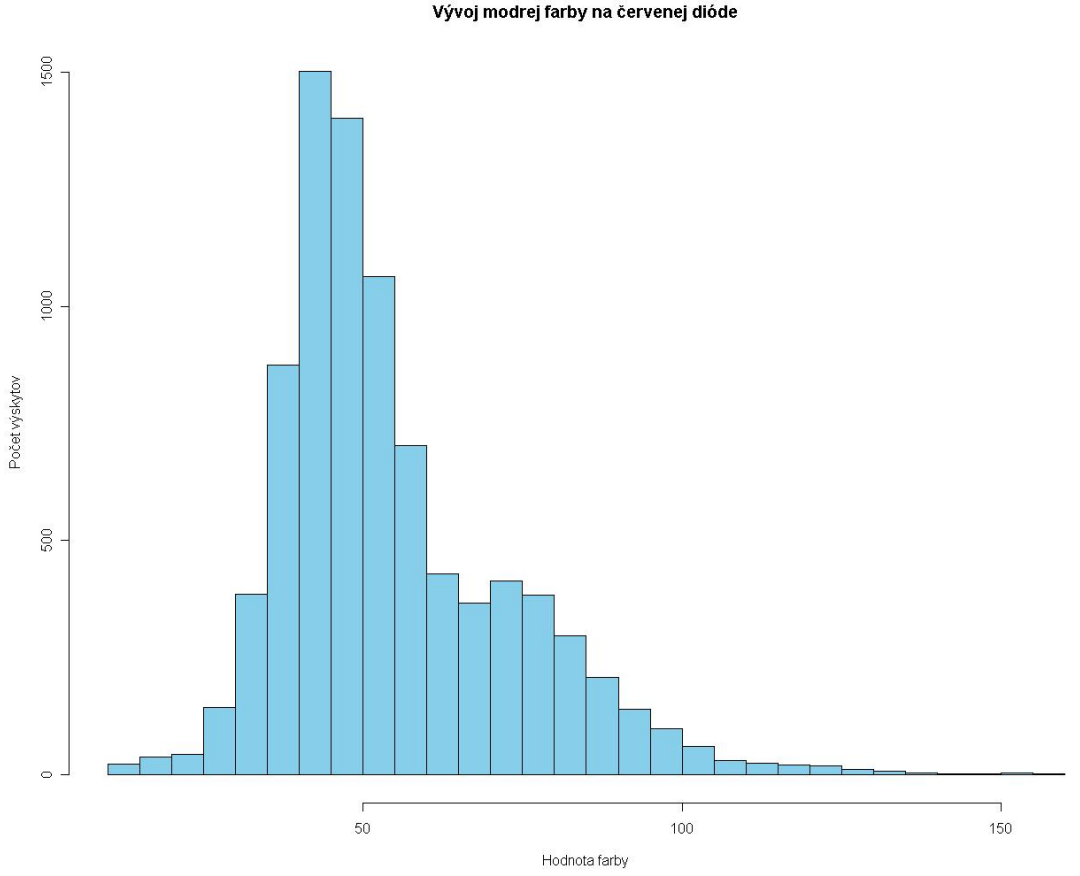
Zoznam obrázkov

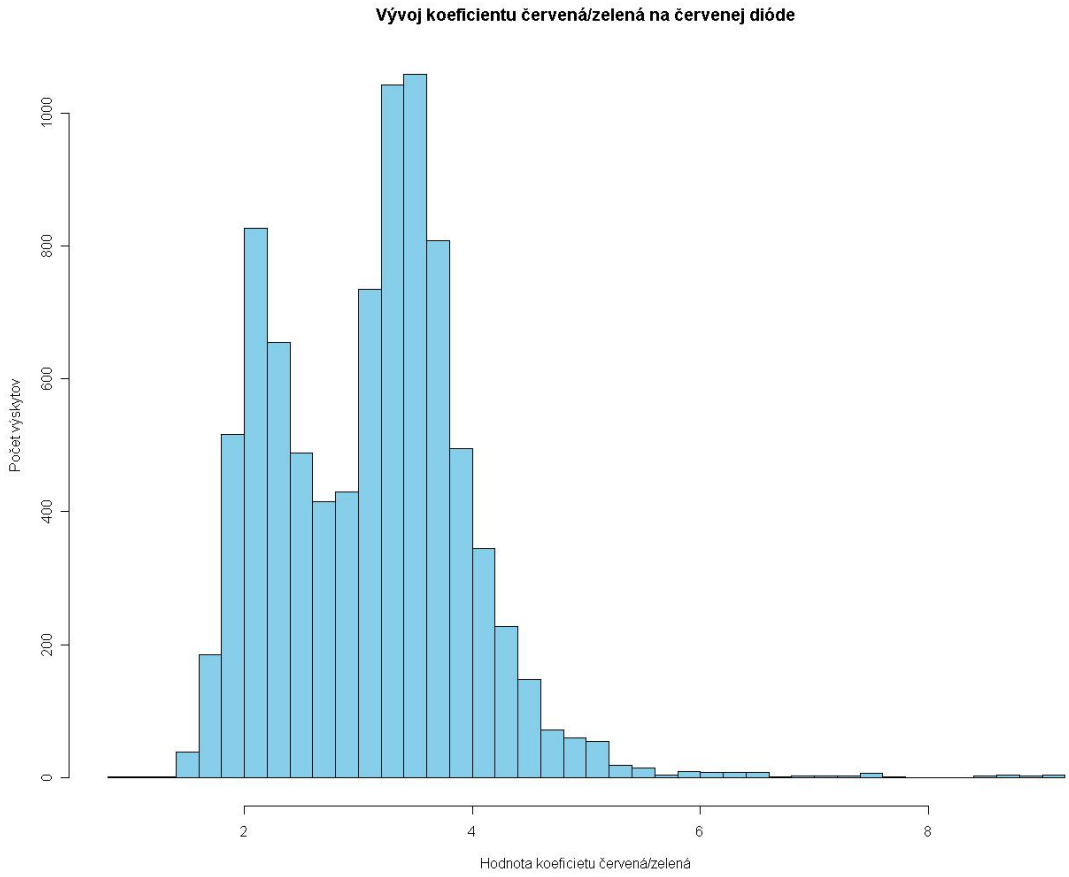
Obrazová príloha grafov farebnosti diód

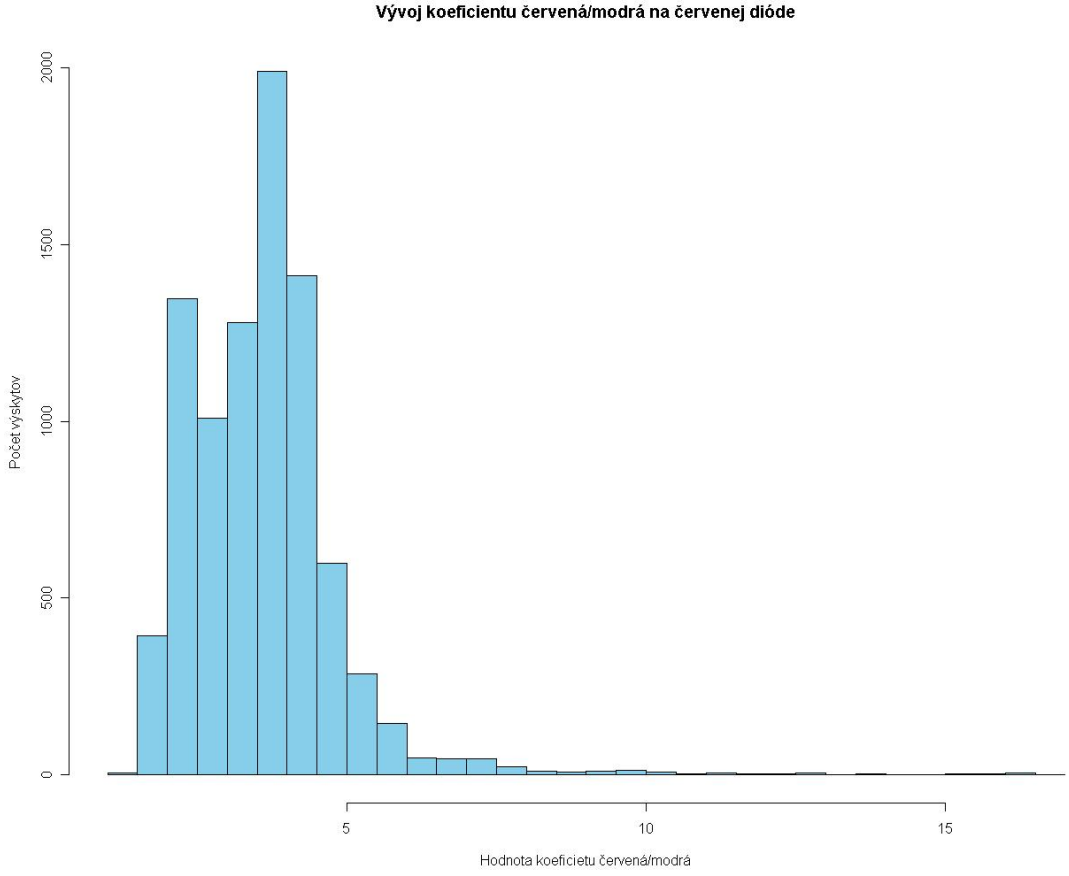
Nasledujúce grafy zobrazujú farebnú situáciu na každej z diód. Spôsob výroby a použitia týchto štatistík je popísaný v kapitole 2.5 Celkovo je pre každú diódu zobrazených sedem grafov. V nadpise grafu sa nachádzajú informácie o meranej veličine a príslušnosti k dióde. Veličinou môže byť niektorá z farebných zložiek farby (červená, zelená, modrá), pomer dvoch zložiek farby (červená/zelená, červená/modrá, zelená/modrá), alebo celkový pomer všetkých troch farebných zložiek (červená/zelená/ modrá). Na x-ovej osi sú zobrazované hodnoty jednotlivých veličín. Na y-novej osi sú zobrazované počty výskytov zistených veličín.

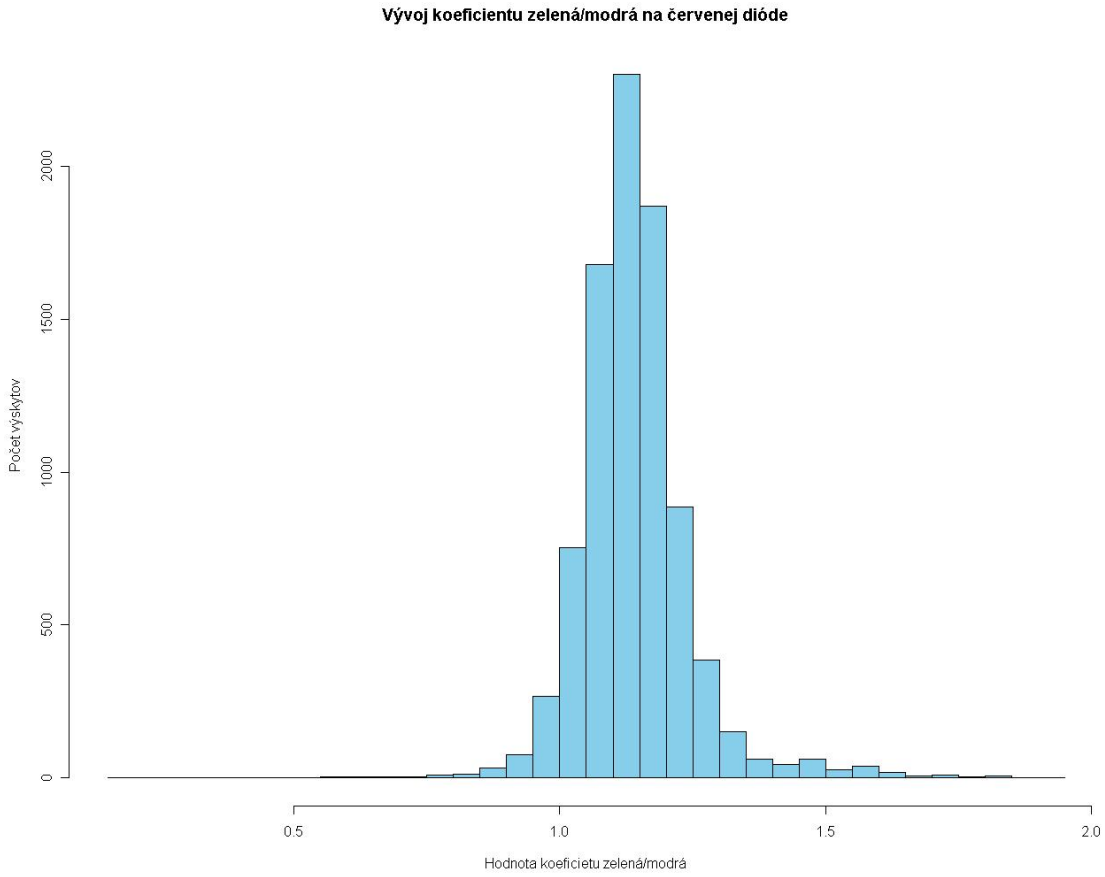


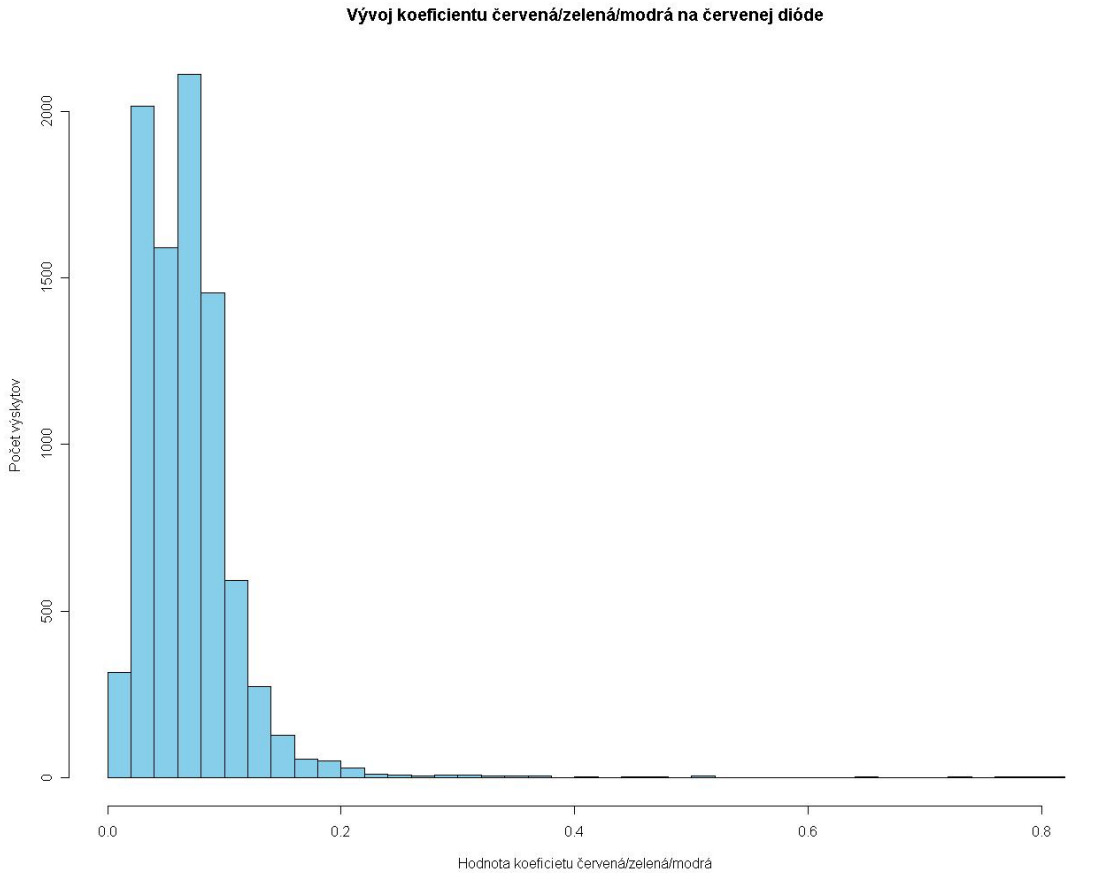

Vývoj zelenej farby na červenej dióde
Počet výskytov
0
500
1000
20
40
60
80
100
120
140
160
Hodnota farby

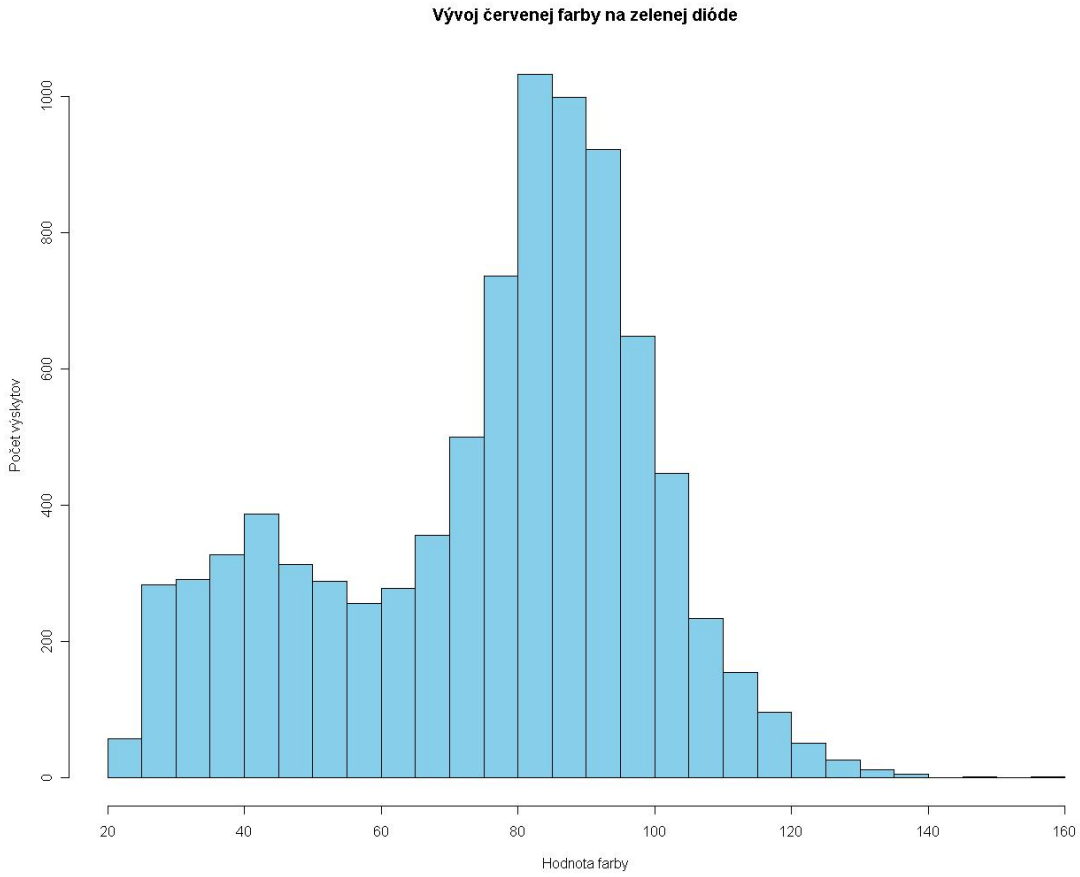

Vývoj modrej farby na červenej dióde
Počet výskytov
0
500
1000
1500
50
100
150
Hodnota farby

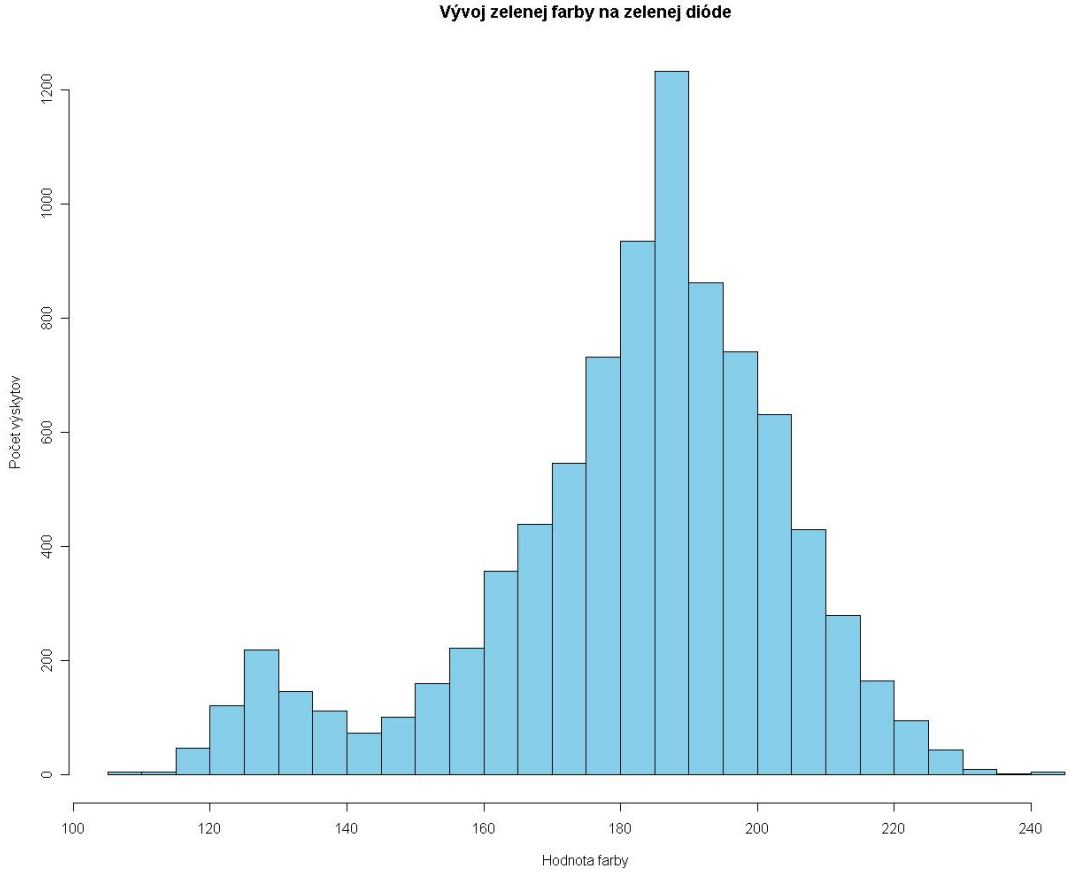

Vývoj koeficientu červená/zelená na červenej dióde
Počet výskytov
0
200
400
600
800
1000
2
4
6
8
Hodnota koeficietu červená/zelená

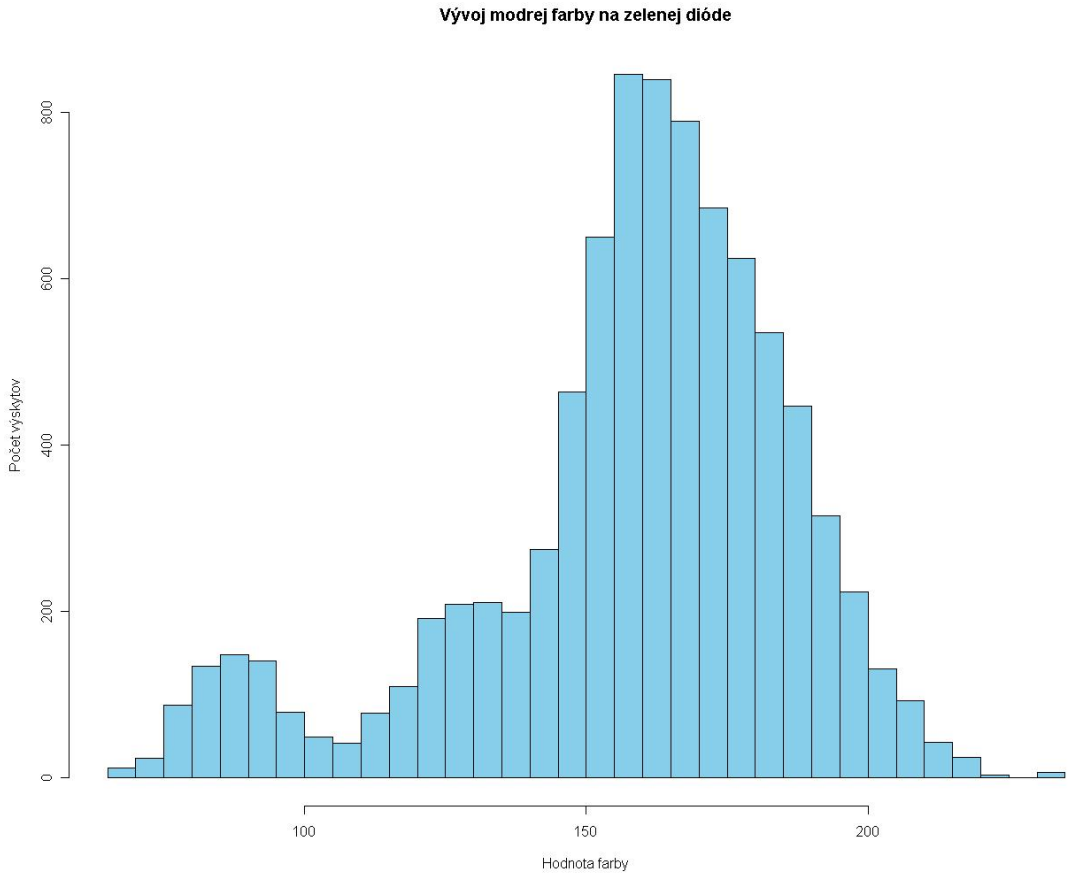

Vývoj koeficientu červená/modrá na červenej dióde
Počet výskytov
0
500
1000
1500
2000
5
10
15
Hodnota koeficietu červená/modrá

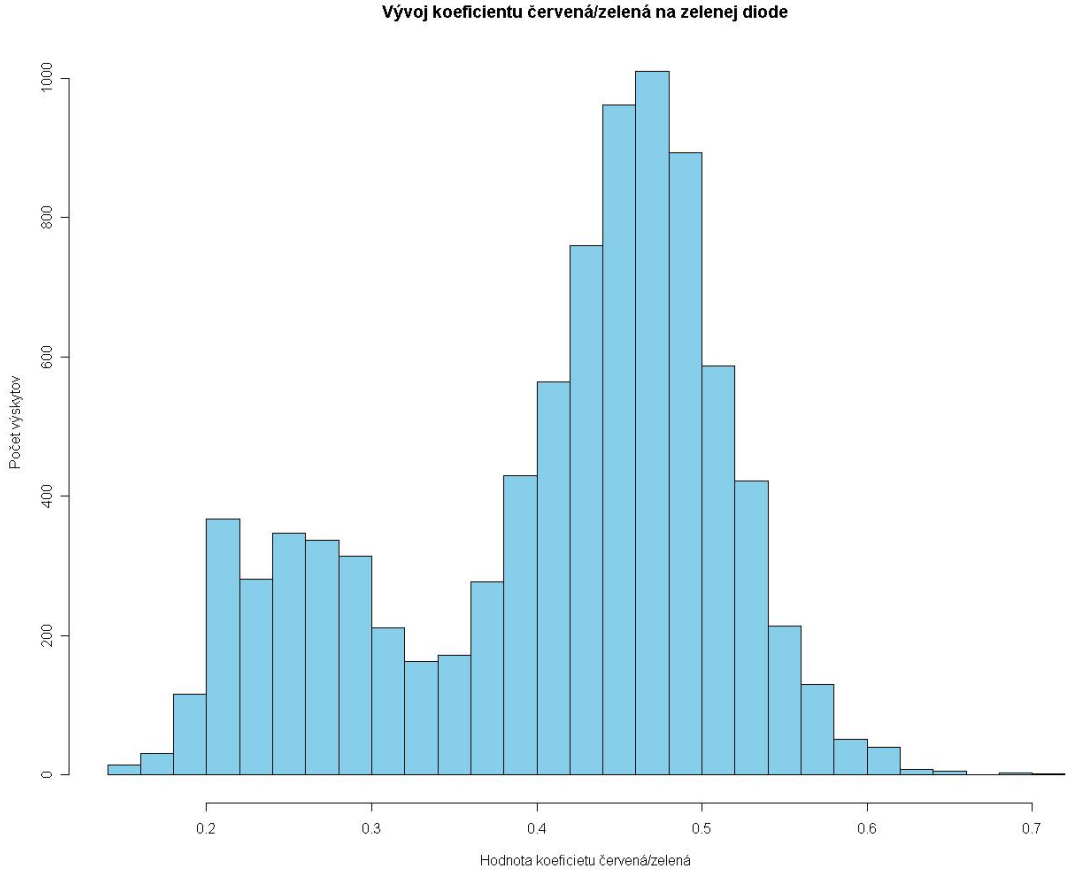

Vývoj koeficientu zelená/modrá na červenej dióde
Počet výskytov
0
500
1000
1500
2000
0.5
1.0
1.5
2.0
Hodnota koeficietu zelená/modrá

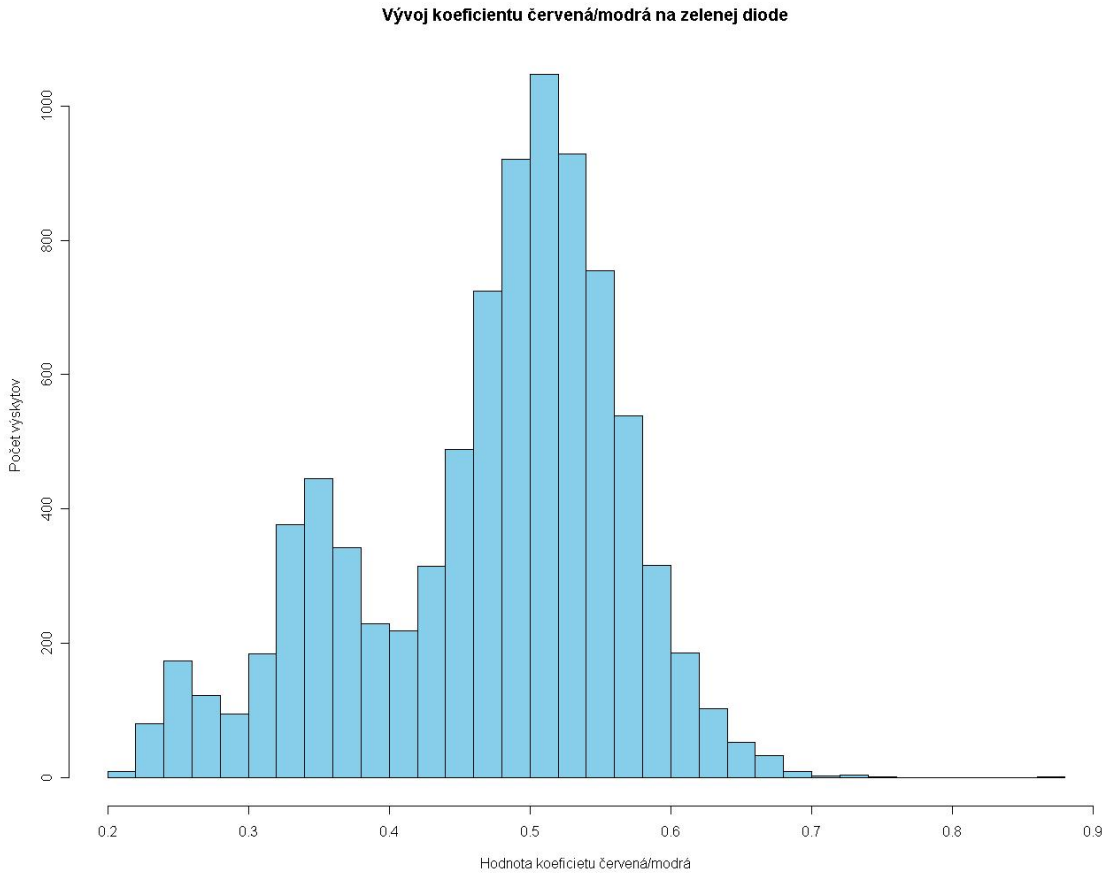

Vývoj koeficientu červená/zelená/modrá na červenej dióde
Počet výskytov
0
500
1000
1500
2000
0.0
0.2
0.4
0.6
0.8
Hodnota koeficietu červená/zelená/modrá

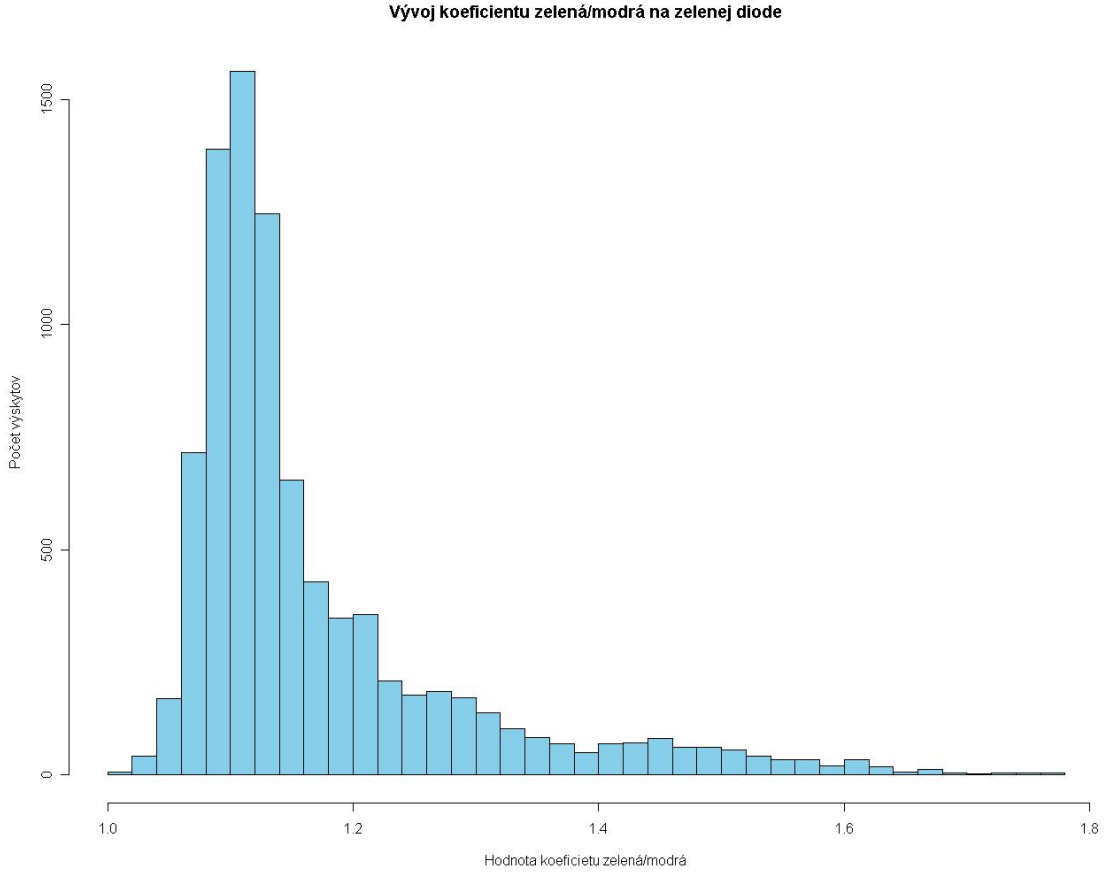

Vývoj červenej farby na zelenej dióde
Počet výskytov
0
200
400
600
800
1000
20
40
60
80
100
120
140
160
Hodnota farby

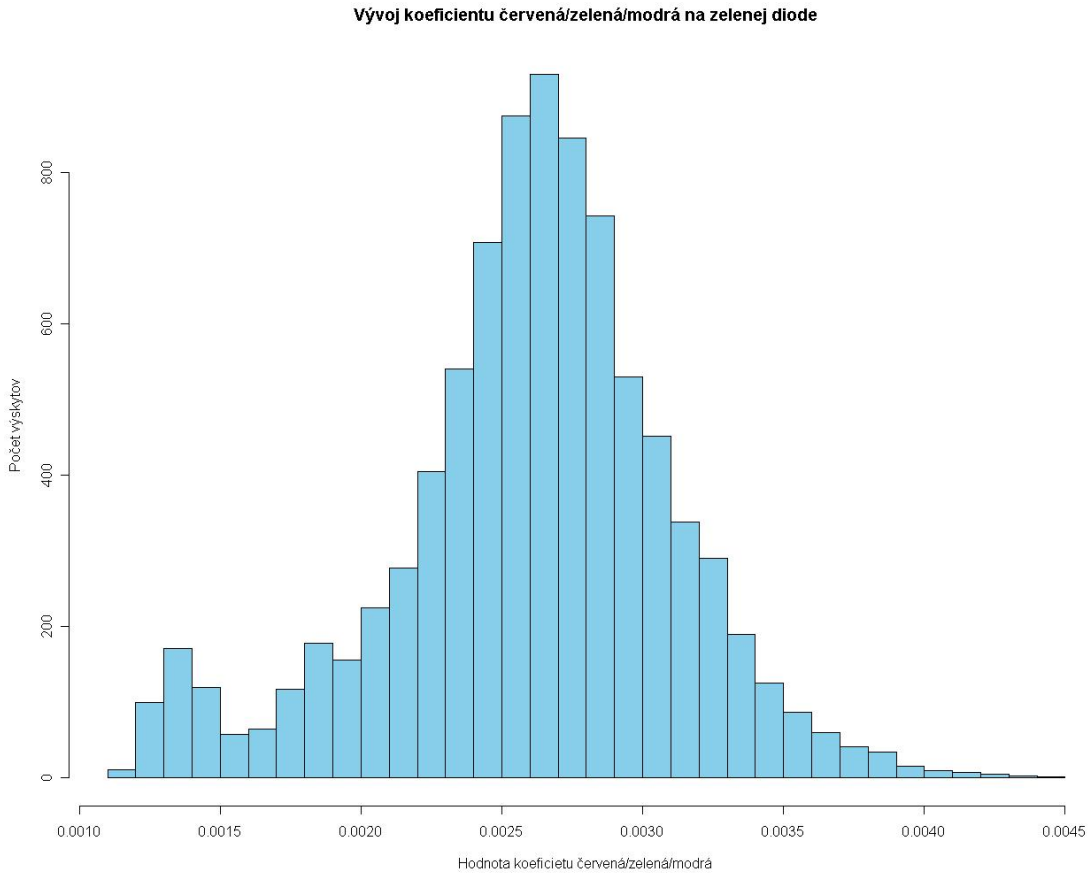

Vývoj zelenej farby na zelenej dióde
Počet výskytov
0
200
400
600
800
1000
1200
100
120
140
160
180
200
220
240
Hodnota farby

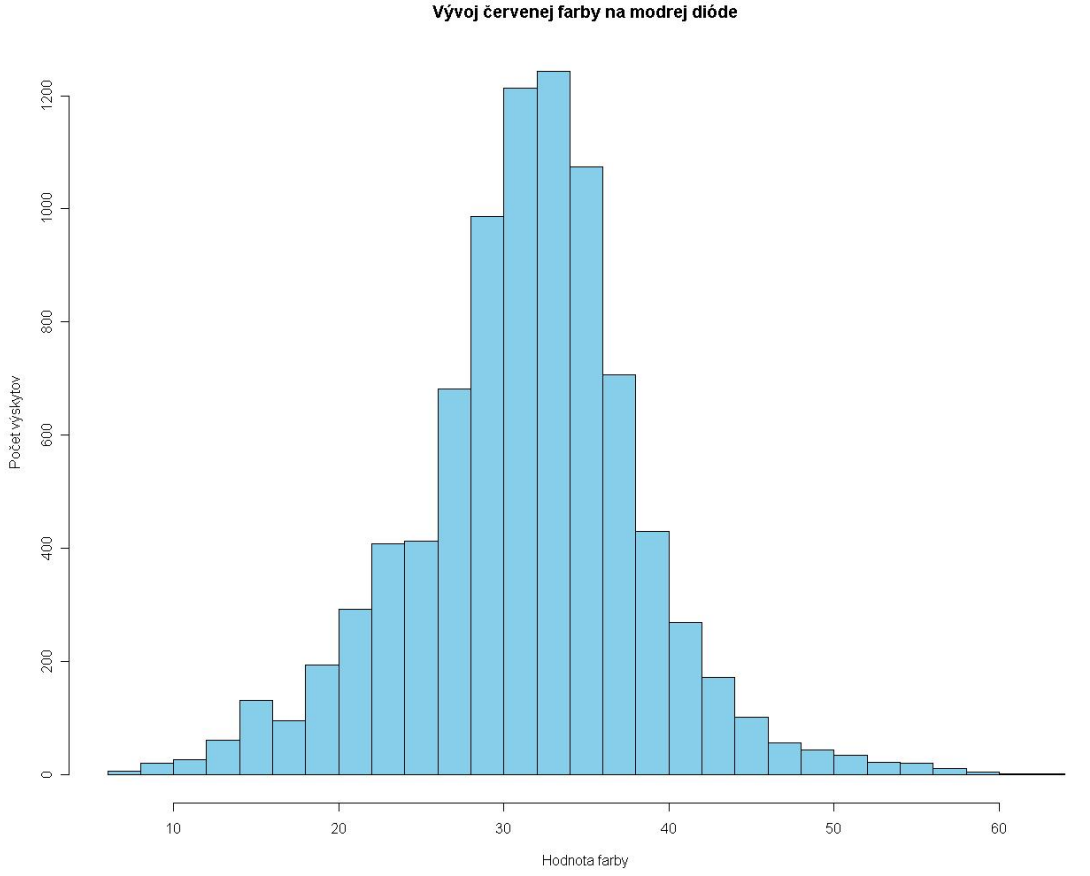

Vývoj modrej farby na zelenej dióde
Počet výskytov
0
200
400
600
800
100
150
200
Hodnota farby

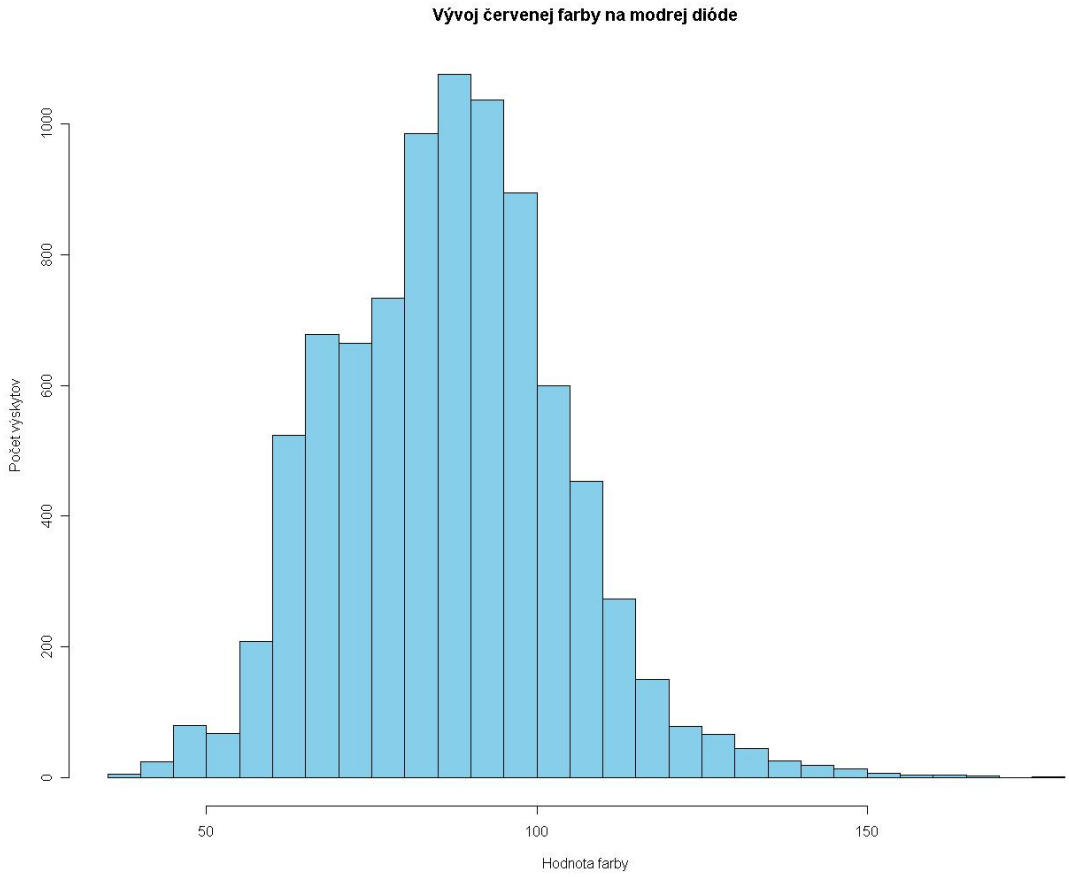

Vývoj koeficientu červená/zelená na zelenej diode
Počet výskytov
0
200
400
600
800
1000
0.2
0.3
0.4
0.5
0.6
0.7
Hodnota koeficietu červená/zelená

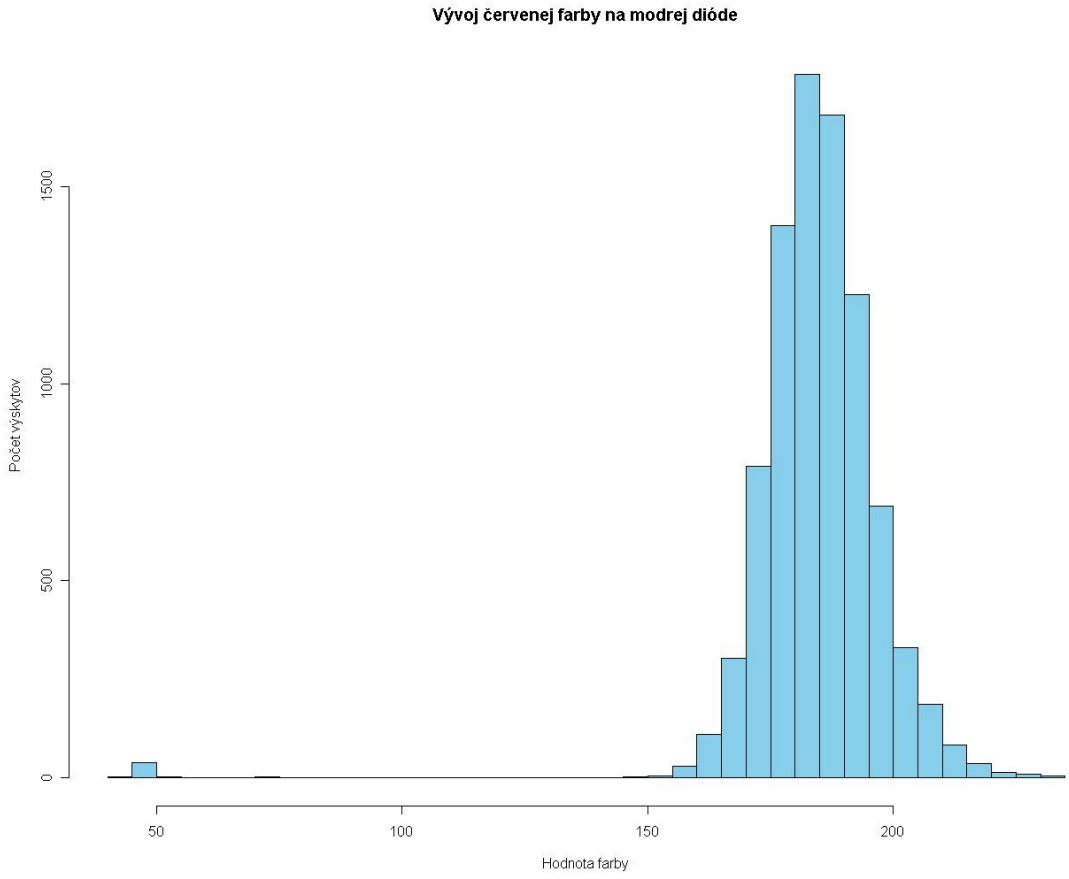

Vývoj koeficientu červená/modrá na zelenej diode
Počet výskytov
0
200
400
600
800
1000
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
Hodnota koeficietu červená/modrá

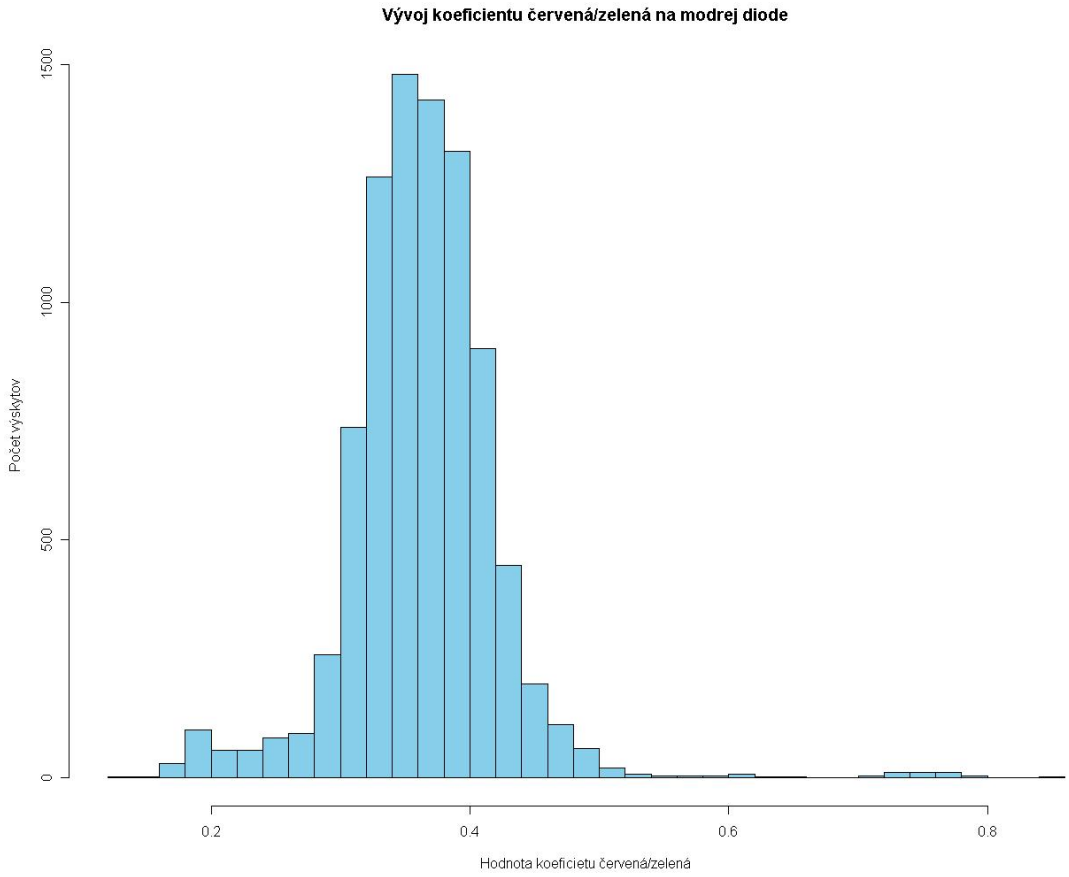

Vývoj koeficientu zelená/modrá na zelenej diode
Počet výskytov
0
500
1000
1500
1.0
1.2
1.4
1.6
1.8
Hodnota koeficietu zelená/modrá

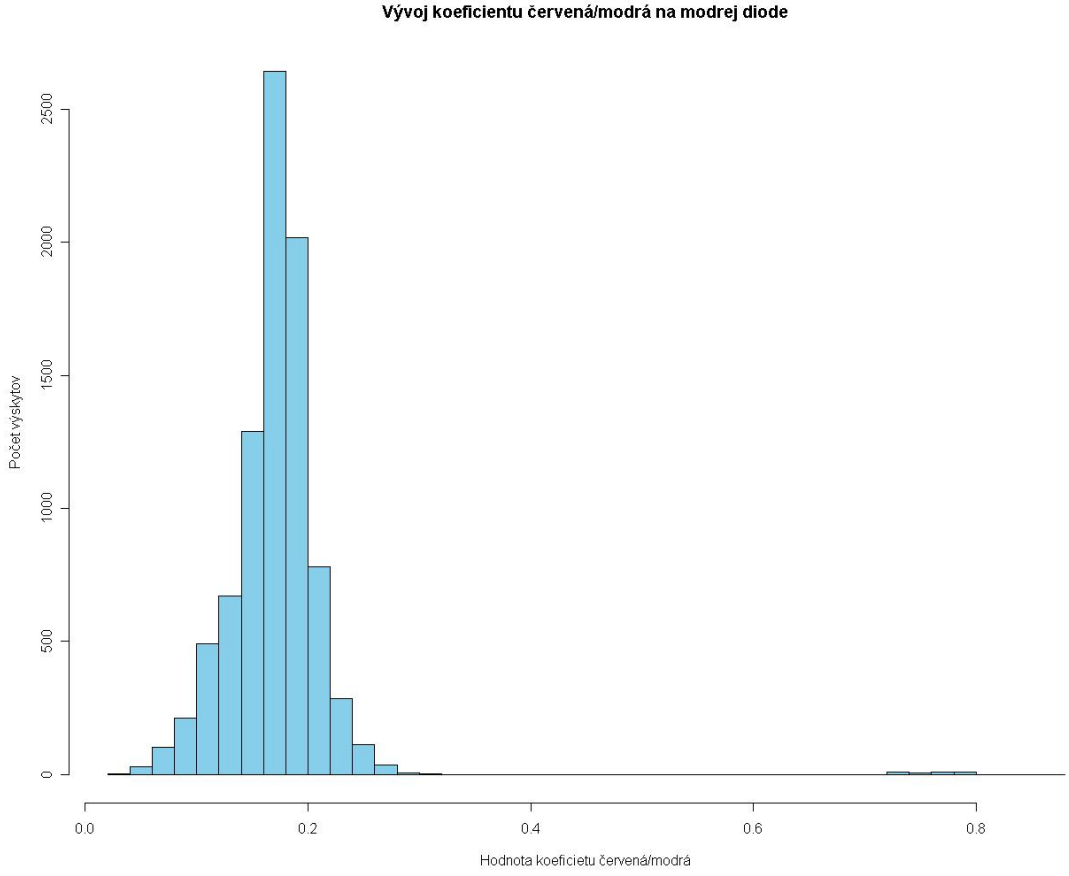

Vývoj koeficientu červená/zelená/modrá na zelenej diode
Počet výskytov
0
200
400
600
800
0.0010
0.0015
0.0020
0.0025
0.0030
0.0035
0.0040
0.0045
Hodnota koeficietu červená/zelená/modrá

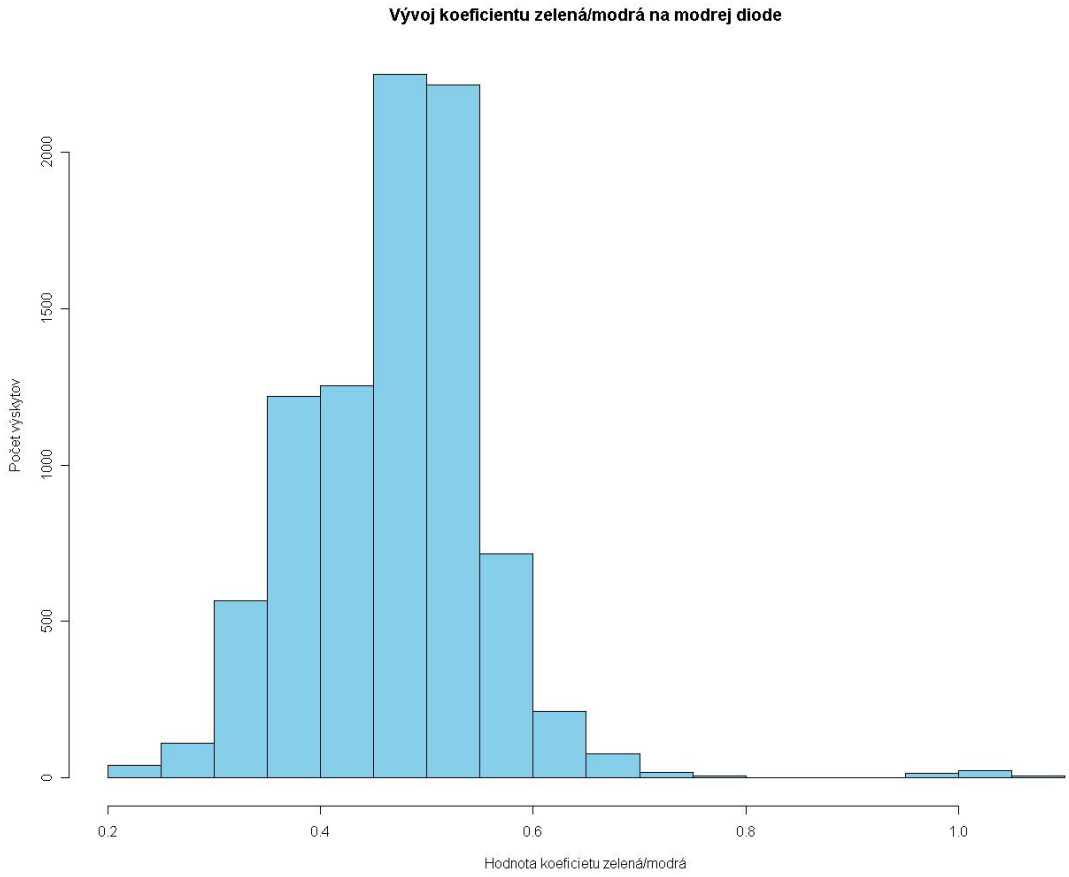

Vývoj červenej farby na modrej dióde
Počet výskytov
0
200
400
600
800
1000
1200
10
20
30
40
50
60
Hodnota farby

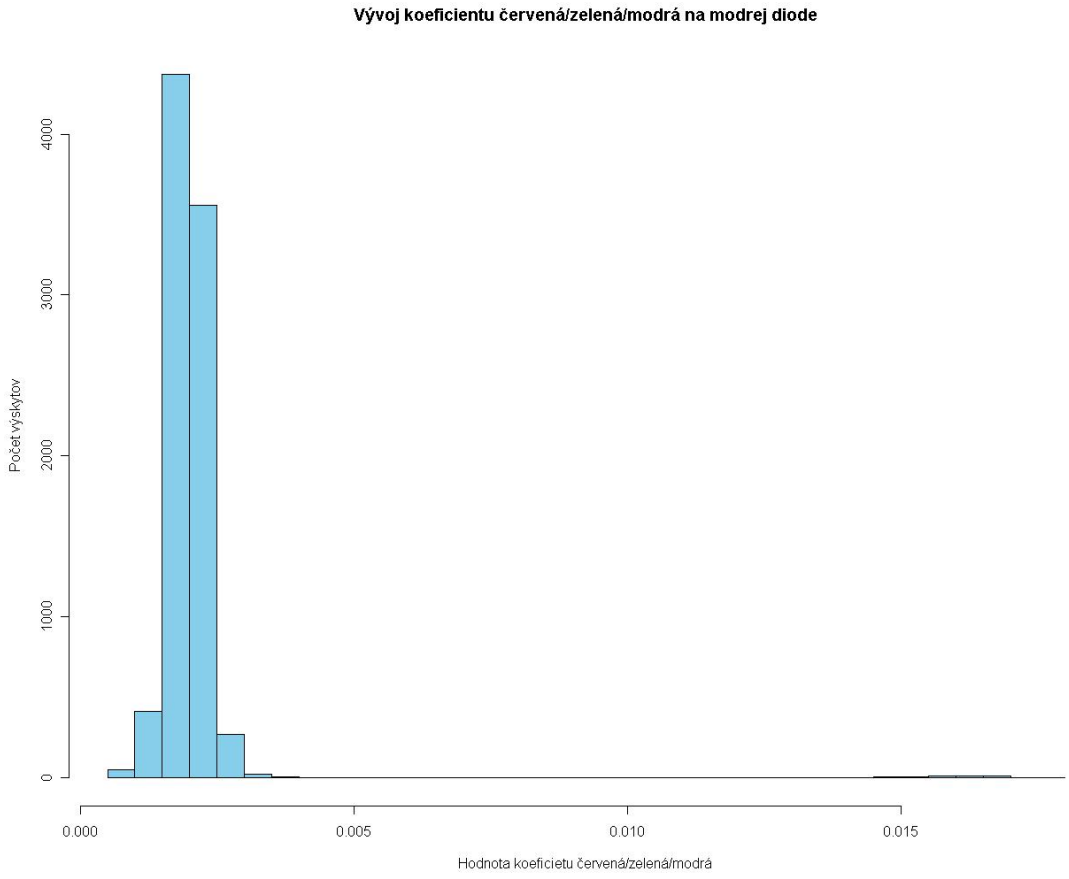

Vývoj červenej farby na modrej dióde
Počet výskytov
0
200
400
600
800
1000
50
100
150
Hodnota farby


Vývoj červenej farby na modrej dióde
Počet výskytov
0
500
1000
1500
50
100
150
200
Hodnota farby


Vývoj koeficientu červená/zelená na modrej diode
Počet výskytov
0
500
1000
1500
0.2
0.4
0.6
0.8
Hodnota koeficietu červená/zelená


Vývoj koeficientu červená/modrá na modrej diode
Počet výskytov
0
500
1000
1500
2000
2500
0.0
0.2
0.4
0.6
0.8
Hodnota koeficietu červená/modrá


Vývoj koeficientu zelená/modrá na modrej diode
Počet výskytov
0
500
1000
1500
2000
0.2
0.4
0.6
0.8
1.0
Hodnota koeficietu zelená/modrá


Vývoj koeficientu červená/zelená/modrá na modrej diode
Počet výskytov
0
1000
2000
3000
4000
0.000
0.005
0.010
0.015
Hodnota koeficietu červená/zelená/modrá